Издательство «Знание»

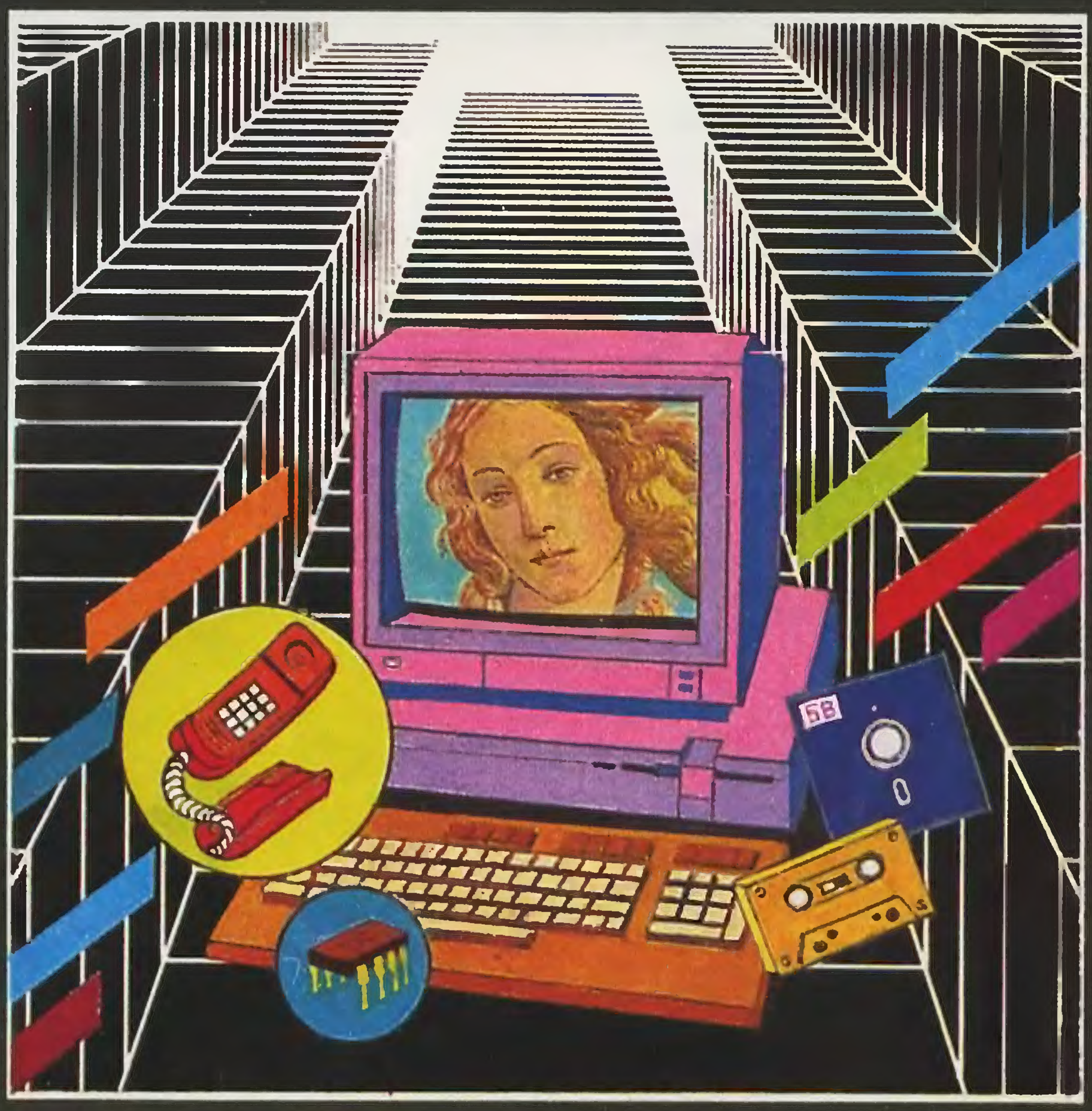
ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Новое в жизни, науке, технике

Подписная научно-популярная серия

Издается ежемесячно с 1988 г.

Что взамен структуры фон-Неймана?



1989

5

Новое
в жизни,
науке,
технике

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Подписная
научно-
популярная
серия

5/1989

Издается
ежемесячно
с 1988 г.

ЧТО ВЗАМЕН СТРУКТУРЫ ФОН НЕЙМАНА?

В НОМЕРЕ:

3 Б. А. БАБАЯН
АРХИТЕКТУРА СУПЕР-ЭВМ

8 И. А. АПОКИН, Д. А. ПОСПЕЛОВ
ПРОЕКТЫ 80-Х ГОДОВ: ПУТЬ К МАШИНАМ ПЯТОГО ПОКОЛЕНИЯ

23 С. В. СЕМЕНИХИН
ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СУПЕР-ЭВМ

40 О. Я. БОКСЕР, А. Г. ВАСИЛЬЧЕНКО, А. А. ПРИЯТКИН
ГОДЕН, НЕ ГОДЕН (НУЖНОЕ ПОДЧЕРКНУТЬ)

РУБРИКИ:

БК за рога

Справки, сообщения, реклама

Издательство
«Знание»
Москва
1989

ББК 32.97
Ч80

Авторы выпуска

БАБАЯН Борис Арташесович — член-корреспондент АН СССР, доктор технических наук, профессор. Лауреат Ленинской и Государственной премий СССР. Крупный специалист в области создания архитектуры супер-ЭВМ

АПОКИН Игорь Алексеевич — кандидат технических наук, старший научный сотрудник Института истории естествознания и техники АН СССР.

ПОСПЕЛОВ Дмитрий Александрович — доктор технических наук, профессор, заведующий сектором ВЦ АН СССР, специалист в области теории автоматов, вычислительных систем и искусственного интеллекта.

СЕМЕНИХИН Сергей Владимирович — доктор технических наук. Специалист в области математического обеспечения ЭВМ. Лауреат Ленинской премии СССР.

БОКСЕР Оскар Яковлевич — лауреат премии Совета Министров СССР, доктор медицинских наук, профессор, заслуженный изобретатель РСФСР. Имеет 4 монографии, свыше 200 статей по физиологии, авиакосмической медицине и биологической кибернетике.

ВАСИЛЬЧЕНКО Александр Георгиевич — старший преподаватель Шуйского государственного педагогического института.

ПРИЯТКИН Андрей Аркадьевич — инженер-программист информационно-вычислительной службы Шуйского государственного педагогического института.

ГВОЗДЕВ Сергей Владимирович — старший научный сотрудник НИИ физики твердого тела Латвийского Государственного университета им. П. Стучки, имеет свыше 30 публикаций в области автоматизации физического эксперимента. Лауреат бронзовой медали ВДНХ.

ЭРНЕСТОНС Гинтс Алдович — инженер-программист НИИ физики твердого тела Латвийского Государственного университета им. П. Стучки, специализируется в области использования персональных ЭВМ в информатике и автоматизации физического эксперимента.

Редактор Б. М. ВАСИЛЬЕВ

ISBN 5-07-000483-2

©Издательство «Знание», 1989 г.

На заре развития суперкомпьютеров их быстродействие измерялось тысячами флопов (флоп — число операций с плавающей запятой в 1 с). Сегодня — в гигафлопах. Завтра этот показатель будет измеряться терафлопами (триллионы операций в 1 с). Один такой суперкомпьютер будет эквивалентен 10 млн. ПВВМ.

Б. А. БАБАЯН

АРХИТЕКТУРА СУПЕР-ЭВМ

Супер-ЭВМ — это машины предельной производительности на данном этапе развития вычислительной техники. Никакого другого определения этого понятия не существует. Однако совершенно очевидно, что те параметры, благодаря которым машина называется супер-ЭВМ в настоящее время, через несколько лет будут у машин средней производительности. Почему же такого рода машины выделены и им даже присвоено специальное название? Это происходит, по крайней мере, по двум причинам. Во-первых, супер-ЭВМ позволяют людям производить расчеты, которые нельзя было выполнить до этого. В результате можно получать новые знания о явлениях, ранее человечеству не известных. Такое положение можно сравнить с открытием новых законов и явлений природы. Супер-ЭВМ в буквальном смысле помогают делать открытия. Во-вторых, разработка предельных по характеристикам машин приводит к необходимости совершенствования методов создания и самой структуры машины. Полученные при этом достижения впоследствии с успехом используются и при разработке других более медленных машин.

В связи с приведенным выше определением супер-ЭВМ нельзя говорить о каких-то их специфических особенностях во все времена. Можно лишь говорить об особенностях или проблемах современных супер-ЭВМ. В следующий период для супер-ЭВМ будущего эти проблемы будут решены, но возникнут другие, актуальные для создания супер-ЭВМ нового поколения. При разработке машины предельной производительности все должно быть предельным.

В статье рассматриваются важные вопросы структуры, или, как говорят, логического построения супер-ЭВМ.

В определенном смысле ЭВМ подобна алгоритму решения большой задачи. Сколь бы ни была сложна задача, она разбивается на совокупность элементарных операций над числами (сложения, умножения и т. д.), которые, в свою очередь, сводятся к совсем уж элементарным таблицам умножения и сложения цифр. Элементарные операции носят совершенно универсальный характер и не специфичны для какой-либо конкретной задачи. Конкретность задачи выражается во взаимосвязи операций, в связи между операциями по исходным данным и результатам. Аналогично любая ЭВМ строится из элементарных логических кирпичиков — операций, обрабатывающих простейшие данные, которые присутствуют в ней в большом количестве. Индивидуальность каждой машины — это конкретный способ или схема соединения между собой ее узлов. Упомянутые элементарные логические операции обладают важным свойством, используя их и соединяя нужными способами, можно построить любую достаточно сложную функцию, т. е. эта система элементарных функций универсальна. Практика разработки машин привела конструкторов к убежденности, что среди большого числа возможных систем элементарных функций следует выбирать простейшие. Поэтому в настоящее время все машины, в том числе и супер-ЭВМ, строятся на базе двоичной арифметики, т. е. элементарными цифрами в них являются нуль и единица. Ясно, что проще нельзя. А в качестве элементарных логических кирпичиков функций выбраны двухместные функции, т. е. функции с двумя входами и одним значением этих функций. В общем подсчете их немного — всего 16, но полезных много меньше.

Физическая реализация этих элементарных функций и методы их связи меняются в машинах от поколения к поколению. Скорость работы машины, конечно, зависит от скорости работы каждого элемента, от скорости передачи

информации от элемента к элементу и от свойств общей логической схемы соединения элементов, в частности, от степени параллельности работы элементов. Именно об этом мы поговорим здесь более подробно.

Чтобы понять проблемы структуры супер-ЭВМ настоящего времени, полезно взглянуть на тот путь, который прошла структура и архитектура ЭВМ. Первые машины имели достаточно простую структурную схему, которая состояла из четырех крупных блоков или устройств: арифметического устройства, устройства ввода-вывода, запоминающего устройства и устройства управления.

В арифметическом устройстве выполнялись операции, исходные данные и промежуточные результаты хранились в запоминающем устройстве. В этом же запоминающем устройстве хранятся команды, которые управляют работой машины. Приблизительно такую структуру имела первая отечественная быстродействующая вычислительная машина, созданная под руководством родоначальника советской вычислительной техники академика С. А. Лебедева в Институте точной механики и вычислительной техники АН СССР.

Команды в этой машине имели приблизительно следующий вид:

Команда	**Адрес**	**Адрес**	**Адрес**
арифметической операции (+, −, х и т. д.)	первого операнда в запоминающем устройстве	второго операнда в запоминающем устройстве	результата операции

Последовательность таких команд считывается из запоминающего устройства, и каждая команда выполняется следующим образом. Из ячеек запоминающего устройства по адресам числа, указанным в команде, считываются операнды для текущей операции, затем эта операция выполняется в арифметическом устройстве и ее результат помещается в запоминающее устройство по адресу, указанному в команде.

Временную диаграмму работы такой машины можно представить себе следующим образом: первая команда (I, II, ОП, Р), где I — считывание первого операнда из запоминающего устройства, II — считывание второго операнда из запоминающего устройства, ОП — выполнение операции в арифметическом устройстве, Р — запись результата в запоминающее устройство. Затем эта последовательность действий машины будет повторена для второй, третьей и т. д. команд. Отсюда хорошо видно, что все блоки в этой машине работают последовательно, ничего не работает в параллель. Сначала считываются числа одно за другим, затем выполняются операции, и далее результат записывается в память. К этому надо еще добавить, что, когда начинает работать ввод или вывод данных, прекращаются арифметические операции. Таким образом, единственный блок в этих машинах, который работал параллельно с основными устройствами, — это устройство управления. По своему характеру устройство управления должно работать параллельно, только так можно организовать работу основных устройств. Отсутствие параллельности в первых машинах было обусловлено по меньшей мере двумя факторами. Во-первых, большие физические размеры и низкая надежность оборудования не позволяли создавать сложной конструкции. Тем более нельзя было ставить вопрос о введении нескольких однотипных, параллельно работающих устройств. Во-вторых, люди просто не имели опыта организации параллельного счета и параллельной работы устройств, а это, как показало дальнейшее развитие, оказалось очень просто.

В дальнейшем по мере прогресса технологии и сокращения размеров элементов, повышения их надежности в машинах все больше и больше стали использовать параллельные структуры и прогресс в этой области на каждом этапе развития происходил именно в классе супер-ЭВМ своего времени. Вместе с повышением миниатюризации

элементов росла их скорость. Этот рост можно очень грубо оценить как линейный. Однако с ростом миниатюризации и надежности конструкторы получили возможность в одной машине использовать значительно больше элементов. Неизмеримо возросла интеграция элементов. Рост числа элементов в процессе миниатюризации пропорционален уже не первой степени, а квадрату, даже кубу сокращения линейных размеров. Это позволяло надеяться при параллельной работе на столь же большое возрастание скорости работы ЭВМ. И конструкторы стали активно искать пути создания параллельных структур машины. Уже следующие варианты конструкций машин позволяли устройству ввода-вывода работать параллельно с арифметическим устройством. Кроме того, появилась возможность обеспечивать параллельную работу запоминающего и арифметического устройств.

Таким образом, все четыре устройства, входящие в конструкцию первых машин, оказались работающими параллельно.

Дальнейший рост интеграции позволил в одной машине использовать несколько однотипных устройств, работающих в параллель. Пожалуй, впервые это произошло с запоминающим устройством. Стали использовать несколько блоков запоминающих устройств, в которых равномерно размещалось все адресное пространство памяти. Стали говорить о расслоении памяти.

Теперь многие обращения в память выполнялись параллельно и сокращались по суммарному времени. Если при этом обеспечить более быстрое арифметическое устройство, то можно значительно ускорить работу машины.

Затем стали вводить несколько арифметических устройств. Эти устройства были специализированы по операциям. Отдельно стал делаться умножитель, отдельно сумматор, отдельно делитель и т. д. Специализированные устройства работали, естественно, быстрее универсальных, тем более что и внутри арифметических устройств стали использовать параллельные алгоритмы выполнения каждой арифметической операции. Кроме того, различные арифметические устройства стали работать в параллель. Это привело к дальнейшему повышению быстродействия.

Наконец, стали использовать параллельную работу в пределах одной машины нескольких устройств управления и устройств ввода-вывода.

Машина, имеющая несколько устройств управления, может выполнять несколько программ одновременно. Такие машины стали называть многопроцессорными. И вот развитие структуры супер-ЭВМ достигло такого уровня, когда каждый тип устройства представлен несколькими экземплярами и все эти устройства, как однотипные, так и разнотипные, имеют возможность работать одновременно параллельно.

Казалось бы, с точки зрения структур достигнуто совершенство. При параллельной работе увеличение числа устройств по мере совершенствования технологии и повышения интеграции должны привести к пропорциональному росту производительности. Однако все оказалось не так просто.

По мере роста числа устройств росло число вариантов их взаимной коммутации в рамках одной машины. Для иллюстрации приведем две возможные схемы: традиционную многопроцессорную систему и матричную, разработанную в свое время в Иллинойском университете США. Эти схемы отличаются сложностью коммутации. Так, в первой схеме каждое устройство управления соединяется с каждым блоком памяти. В такой структуре сложность коммутации увеличивается пропорционально квадрату чисел устройств, в нее входящих, и исходя из практических соображений, невозможно соединить больше 16 устройств управления, не потеряв значительно в производительности. Казалось, все преимущества на стороне второй схемы, так как увеличение числа параллельно работающих устройств усложняет удельную (на одно устройство) коммутацию. Однако практика показывает, что дело обстоит значительно сложнее. Не надо забывать, что перед создателями супер-ЭВМ стоит задача повышения производительности системы, в том числе при

решении одной мощной задачи. Поэтому структура должна быть такой, чтобы программисты имели возможность написать реальную программу, достаточно эффективно загружающую все параллельно работающие устройства. То, что это не всегда возможно, видно из построения предельного примера. Если, например, мы в качестве структуры сверхпараллельной супер-ЭВМ выберем структуру, состоящую из большого числа никак не связанных между собой более простых машин, то хотя такая структура может формально обеспечить достаточно большое число выполняемых операций в секунду, нельзя даже говорить о возможности решения какой-либо одной большой задачи на такой структуре.

Для того чтобы иметь возможность фактически заставить параллельно работать различные устройства супер-ЭВМ, эта структура должна быть в определенном смысле адекватна структуре задач, которые должны на ней решаться.

Правильный учет структуры задач и методов программирования с самого начала развития вычислительных машин и супер-ЭВМ в частности определял основные направления развития структур ЭВМ и супер-ЭВМ. Удобство программирования — одна из важнейших проблем вычислительной техники, и в этой области, именно в классе супер-ЭВМ, на отечественных машинах достигнуты максимальные успехи. Созданы машины, наиболее удобные в программировании и сокращающие трудозатраты на программирование в несколько раз.

Вообще говоря, проблема программирования является очень сложной и определяющей в развитии ЭВМ и супер-ЭВМ в частности. Дело в том, что по информационной сложности программы для ЭВМ по всей видимости превосходят любой другой задокументированный результат человеческой деятельности. Этот вопрос оказался настолько серьезным, что на границе 60-х и 70-х гг. стали говорить о кризисе программирования, который стал значительным тормозом в развитии средств вычислительной техники. Его приблизительно можно описать следующим образом. Существует достаточное количество крайне полезных и весьма сложных задач, для решения которых достаточна существующая производительность и другие функциональные характеристики разработанных машин, но программу для этих задач невозможно составить существующими методами программирования ввиду большой информационной сложности. При этом увеличение числа программистов не приводит к успеху. Практика программирования с достоверностью приводит к выводу, что важнейшим методом облегчения программирования является использование специально созданных для изложения алгоритмов универсальных языков программирования, называемых языками высокого уровня. Хорошо понимая это, разработчики вычислительных машин стали ориентировать их структуру на эффективную реализацию этих языков. В частности, в некоторых отечественных супер-ЭВМ удалось полностью избавиться от программирования низкого уровня.

Возвращаясь теперь к изложенным выше вариантам структур ЭВМ, отметим, что к важным недостаткам многих из них относится тот факт, что реальные задачи не удавалось программировать для них так, чтобы получить достаточно высокую загрузку параллельно работающего оборудования. Казалось бы, разработчики архитектуры дают возможность использовать высокопараллельный счет и, как следствие, получить большую скорость вычисления, а программисты не могут «достаточно хорошо» запрограммировать задачи. Конечно, здесь беда совсем не в слабости программистов, а в неправильно выбранной структуре машин. При этом речь идет об очень большой потере эффективности. При плохо разработанной структуре производительность системы составляет единицы процентов от предельно достижимого при полной загрузке параллельно работающих устройств.

Безусловно, положительно зарекомендовала себя хорошо известная теперь многопроцессорная архитектура. Однако она не позволяет увеличить число процессоров из-за сложности ком-

мутации. Поиски параллельных структур, адекватных решаемым алгоритмам, обратили внимание конструкторов ЭВМ на векторные вычисления. Создать структуру, на которой можно было бы получить высокопараллельную обработку векторных вычислений, оказалось не так сложно. Сейчас можно смело сказать, что если бы любая задача полностью сводилась к работе над векторами достаточно большой длины, можно было бы разработать систему с высокоэффективной параллельностью. И таких проектов чисто векторных супер-ЭВМ появилось немало. Однако опыт их использования показал, что в любой, даже самой векторной задаче существует достаточно много скалярных вычислений.

В реальных задачах число скалярных вычислений может составлять 30—50%. Таким образом, если все 70% векторных вычислений в задаче методами распараллеливания выполнять практически за нулевое время, то вся задача будет считаться быстрее не более чем в 3 раза. Методы распараллеливания векторного счета стали включать во все существующие супер-ЭВМ. Однако стало ясно, что ключевой будет проблема распараллеливания скалярных вычислений. Именно в этой области в настоящее время ведутся интенсивные поисковые научные работы разработчиков структуры ЭВМ. Существует целый ряд крупномасштабных проектов как у нас в стране, так и за рубежом, направленных на решение этой сложной проблемы.

НЕЙРОКОМПЬЮТЕРИЗАЦИЯ: СОЗДАНИЕ АНАЛОГА ЧЕЛОВЕЧЕСКОГО МОЗГА

Нейрокомпьютер — это ЭВМ, которая может обучаться. Она поступает так же, как человек, т. е. многократно просматривает информацию, делает множество ошибок, учится на них, исправляет и, наконец, успешно справляется с задачей. Нейрокомпьютеры незаменимы там, где беспомощны обычные ЭВМ (распознавание образов, восприятие человеческой речи, рукописного текста и др.).

В настоящее время нейрокомпьютеры выходят на стадии опытных разработок и становятся коммерческими. Ряд фирм — Hecht — Nielsen Neurocomputer Corp. (HNC), IBM Corp., Science Application International Corp. (SAIC), Texas Instruments Corp., TRW Inc. — занимаются их разработкой.

Функционирование существующих ЭВМ основано на принципе фон Неймана, поэтому любая задача должна быть подготовлена в виде четного алгоритма. В то же время существует большое количество задач, для которых подготовка алгоритма очень сложна или вообще невозможна. Эти задачи характеризуются тремя общими свойствами: известен способ их решения; можно привести множество примеров; каждая задача имеет похожие объекты в двух множествах.

Нейрокомпьютеризацию можно назвать инженерной дисциплиной, которая имеет дело с программируемыми информационными системами. Вместо использования алгоритма нейросеть создает свои собственные правила посредством анализа различных результатов и примеров.

Нейрокомпьютеризация является относительно новым направлением и первым альтернативным вариантом алгоритмического программирования. Однако нейрокомпьютеры не заменяют обычные ЭВМ и не составляют им конкуренции, так как служат для решения принципиально иных задач.

Нейрон человеческого мозга с ≈ 10 000 других нейронов, причем имеются сильные и слабые связи. При получении какой-либо информации нейроны обмениваются сигналами.

Нейросеть, создаваемая инженерами, слабо напоминает биологическую систему, поскольку в настоящее время слишком мало известно о работе мозга. Однако нейровычисления уже могут дать некоторые результаты, недостижимые при алгоритмическом программировании.

Нейросеть содержит множество процессорных элементов, которые имеют много входов и один выход, подсоединяемый к другим процессорным элементам. Каждый процессорный элемент первого слоя получает сигнал от входного массива. Обрабатывая данные в соответствии с заложенной функцией, процессорный элемент первого слоя передает сигнал на следующий слой. Результатом является выходной массив, который по некоторым характеристикам похож на входной. Так как адаптационные коэффициенты (веса) могут меняться во времени, сеть обучается.

Каждый процессорный элемент обычно содержит свою небольшую память, хранящую результаты предыдущих вычислений и коэффициенты передаточной функции. Часто нейросеть делится на уровни, когда элементы одного уровня

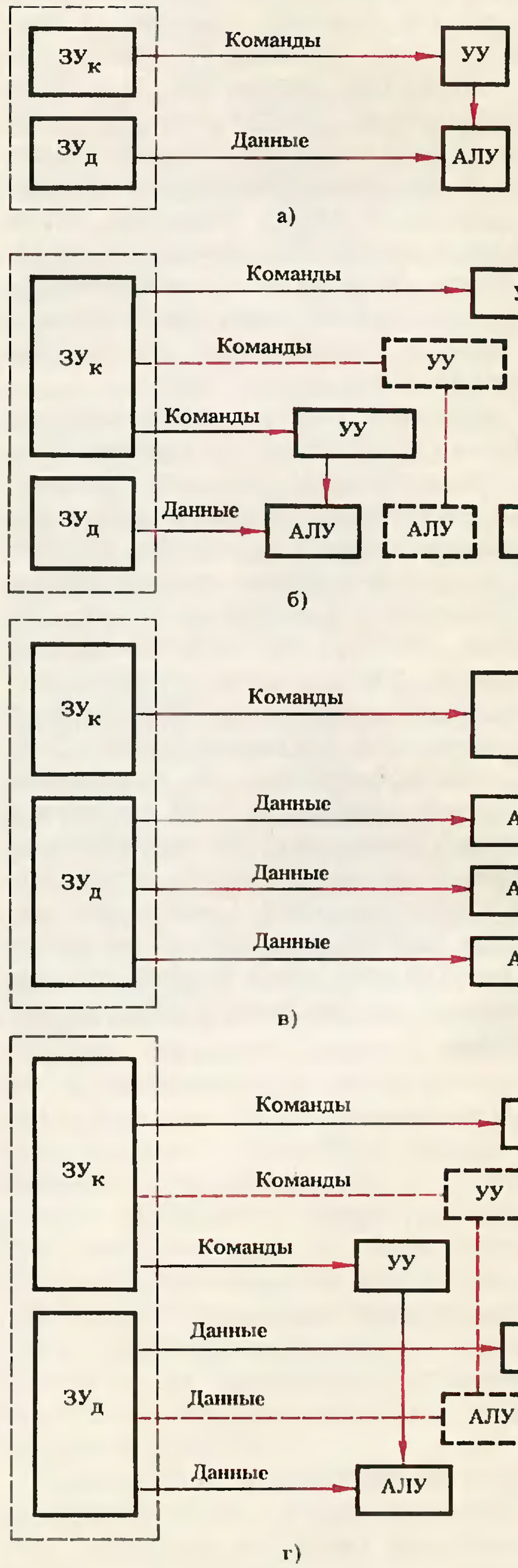
(Продолжение на стр. 22).

В настоящее время завершается целый этап в истории ЭВМ — заканчивается эра IBM PS — совместимых ПЭВМ. Одновременно начался интенсивный процесс внедрения 32-разрядных компьютеров, среди которых наибольший интерес вызывают машины на базе PISC-процессоров и транспьютеров.

И. А. АПОКИН,
Д. А. ПОСПЕЛОВ

ПРОЕКТЫ 80-Х ГОДОВ: ПУТЬ К МАШИНАМ ПЯТОГО ПОКОЛЕНИЯ

Практически все существующие сегодня структуры высокопроизводительных ЭВМ сводятся к четырем классам их функциональной организации: ОКОД — с одиночным потоком команд и одиночным потоком данных (рис. 1, *а*); МКОД — с множественным потоком команд и одиночным потоком данных (рис. 1, *б*); ОКМД — с одиночным потоком команд и множественным потоком данных (рис. 1, *в*); МКМД — с множественным потоком команд и множественным потоком данных (рис. 1 *г*). Однако для характеристики проектов машин пятого поколения этой классификации недостаточно, поскольку большинство этих проектов относится к четвертому типу, т. е. с множественными потоками команд и данных и соответственно имеют многопроцессорную структуру. При этом подходы к реализации этой структуры весьма различны. По состоянию на середину 80-х годов только в США выполнялось около 150 различных по архитектуре проектов ЭВМ (или отдельных процессоров), которые в принципе могли бы стать основой серийных систем пятого поколения. Эти проекты выполнялись в основном академической наукой (преимущественно университетскими исследовательскими центрами), промышленные фирмы участвовали в разработке относительно небольшого числа проектов. Объясняется это тем, что до настоящего времени практически ни один из проектов с радикальными изменениями в архитектуре (по сравнению с существующими ЭВМ) не имеет шансов стать основой выпуска серийных машин.



Потоковые и редукционные машины

Для характеристики проектов ЭВМ пятого поколения может быть использован ряд разработанных в 80-е годы классификаций, из которых в первую очередь отметим классификацию по принципу управления вычислительным процессом, подразделяющую все машины на три группы.

К первой группе относятся машины, управляемые командами. То, что машина должна управляться фиксированной последовательностью команд (в том порядке, в каком они записаны в программе), казалось очевидным до тех пор, пока Дж. Деннис (США) в 1967 г. не предложил альтернативную структуру. По этой структуре должны выполняться все команды (независимо от их места в программе), которые могут быть исполнены (т. е. для которых имеются в наличии операнды) [1]. Иными словами, управление вычислительным процессом переходит к операндам, т. е. данным, или, если угодно, к потоку данных. Машины с такой архитектурой получили название потоковых.

К третьей группе относятся машины, управляемые запросами, т. е. выполнение какой-либо операции определяется сиюминутной потребностью в ее результате. Принципы создания таких машин и языков для них были исследованы в 1971—1974 гг. в работах Дж. Бакуса и К. Берклинга. Машины, управляемые запросами, предназначены для непосредственного выполнения операций, записанных на функциональном языке. В этой связи они получили название редукционных (от английского reduction — сокращение), поскольку работают по принципу сокращения выражений текста программы на функциональном языке. В отличие от потоковых машин, несколько экспериментальных образцов которых к середине 80-х годов находилось в эксплуатации, ни одной редукционной машины к тому времени не было построено. Это связано с трудностями реализации идей функционального программирования.

Остановимся несколько подробнее на развитии идей, лежащих в основе машин с потоковой и редукционной архитектурой и соответствующих проектах.

Идея потоковой машины заключается в создании ЭВМ, способных одновременно выполнять все команды, имеющие операнды. Из этого обстоятельства следует целесообразность наличия многочисленных процессорных элементов. К важному принципу архитектуры потоковых машин относится возможность динамической реконфигурации: в каждый момент времени может быть готово для выполнения различное число команд. Реконфигурация может осуществляться как аппаратным способом (так называемые потоковые IM-машины), так и с помощью программных средств — путем использования специального формата команд (так называемые SM-машины)[1]. За счет возможностей реконфигурации потоковая ЭВМ, являясь универсальной машиной, может в принципе решать задачи с той же производительностью, что и специализированные системы.

Первые работы по потоковым машинам были начаты в Массачусетсском технологическом институте (МТИ) под руководством Дж. Денниса. Сначала Деннис и Д. Майсенас предложили проект элементарного потокового процессора, ориентированного на выполнение некоторых параллельных алгоритмов, встречающихся в задачах обработки сигналов. Программы для этого процессора были написаны на DF-языке, настолько простом, что в нем отсутствовали даже средства для представления условного перехода и цикла.

Следующим этапом развития потоковых ЭВМ стал проект базового потокового процессора, также предложенный Деннисом и Майсенасом. Этот процессор предназначен для выполнения программ, записанных на более сложном алгоритмическом языке — базовом потоком языке Денниса. Работы по этому проекту, проводимые в МТИ, явились, в свою очередь, этапом на пути создания универсальной ЭВМ, использующей потоковый язык высокого уровня. Такой язык, получивший название ВАЛ, был разработан, и по уровню он

[1] Идея создания IM- и SM-машин принадлежит Р. Миллеру и Дж. Коку.

приблизительно соответствует ФОРТРАНу и АЛГОЛу. В первой половине 80-х годов в МТИ была построена модель потоковой ЭВМ, содержащая восемь процессорных элементов. Полный вариант потоковой машины Денниса предполагает значительное увеличение числа процессорных элементов. Использование тысячи элементов позволяет достичь производительности на уровне 1 млрд. оп/с.

В том же МТИ по проекту профессора Арвинда в 80-е годы выполнен ряд работ по созданию потоковой машины, существенно отличающейся по архитектуре от машины Денниса. Фундаментальная проблема, общая для многих потоковых машин, — организация хранения и пересылки данных, особенно матриц данных. Идея Арвинда, предложенная им в середине 70-х годов, состоит в том, что в большинстве случаев данные не пересылаются, а остаются в запоминающих устройствах, доступных для всех процессоров. Пересылаются адреса и команды, следующие путями потоков данных. Архитектура машины Арвинда получила название динамической (в отличие от машины Денниса со статической архитектурой). Аппаратурная реализация динамической архитектуры сложнее. Тем не менее в первой половине 80-х годов в МТИ был построен макет машины с такой архитектурой. Этот макет содержит 64 процессора, аппаратурно реализующих язык ЛИСП (процессоры типа 3600 фирмы «Симболикс», США).

Первая действующая потоковая машина была создана во Франции в исследовательском центре в г. Тулузе. Руководитель проекта — Ж. Сир, один из авторов языка высокого уровня ЛАЮ (язык с единственным присваиванием, т. е. любой переменной в процессе исполнения программы может быть присвоено только одно значение). Модель вычислений, соответствующая принципу единственного присваивания, является потоковой, поскольку управление вычислительным процессом определяется потоком данных. Это же название (ЛАЮ) употребляется по отношению и к проекту в целом, и к конструируемой машине.

Первоначально предполагалось создать специальный процессор для выполнения параллельных вычислений, подсоединяемый к универсальной ЭВМ. В ходе разработки и языка и структуры процессора выяснялась возможность создания универсальной ЭВМ, поскольку язык ЛАЮ позволяет составлять и эффективно выполнять не только параллельные, но и последовательные программы. В первой половине 80-х годов был построен и отлажен прототип машины, содержащий 32 идентичных процессорных элемента, оперирующих со словами длиной 32 бита и взаимосвязанных с помощью шести внешних шин.

Один из наиболее крупных проектов в области потоковых ЭВМ осуществляется в Манчестерском университете (Великобритания), имеющем давние традиции создания машин экстракласса. В середине 70-х годов П. Треливен и Дж. Гард предложили проект универсальной потоковой ЭВМ, которую они назвали динамической параллельной машиной с разделением команд и операндов. Треливен и Гард ввели понятия статических и динамических параллельных систем, связанные со способом выполнения итеративных вычислений. Они назвали динамическими системы, которые позволяют начать $(n+1)$-ю итерацию до окончания n-й итерации. Статические системы лишены такой возможности [2].

Машина Треливена и Гарда представляет собой SM-машину, работающую по принципу интерпретатора без обратной связи. Используемый язык программирования имеет уровень машинного языка. Одна из задач проекта — уменьшение и упорядочение соединительной сети. В 80-е годы проект (с рядом модификаций) осуществляется под руководством Дж. Гарда и И. Уотсона. Опытный образец машины работает с 1981 г. и находится в стадии постоянного совершенствования математического обеспечения. Конечная цель — создание системы с быстродействием до 1 млрд. оп/с.

Первая коммерческая потоковая машина была анонсирована японской фирмой «Ниппон электрик» в 1985 г. Напомним, что разработка потоковой машины предусматривается японским

проектом ЭВМ пятого поколения. В 80-е годы исследования в области потоковых машин проводятся в Японии несколькими фирмами и университетами. Упомянем наиболее известные проекты.

Двумя проектами руководит М. Амамия (фирма NTT). Один из проектов предусматривает создание высокопроизводительной параллельной матрицы процессоров для научных вычислений, а другой — потоковой машины для параллельной обработки списков. Конечная цель разработки матрицы процессоров — достижение производительности 10 млрд. оп/с. Универсальность систем сочетается с некоторой специализацией, обеспечивающей более эффективное выполнение определенных программ. В ходе выполнения проекта разработан функциональный язык высокого уровня ВАЛИД, объединяющий ряд свойств языка ЛИСП и упоминавшегося выше потокового языка ВАЛ. В первой половине 80-х годов была построена экспериментальная модель, получившая название «Эдди». Модель содержит 16 процессорных элементов (матрица 4×4), каждый из которых связан с восемью соседними элементами.

В ходе создания потоковой ЭВМ для обработки списков в первой половине 80-х годов был создан сначала первый вариант макета этой машины (ДФМ-1), а затем второй ее вариант (ДФМ-2). Последний представляет собой систему из восьми процессорных элементов, каждый из которых имеет память для списочных данных (32 тыс. слов) и управляющую память. Создание ДФМ-2 — начальный этап работы по реализации проекта, который в полном объеме предусматривает систему, содержащую 128—256 процессорных элементов. В электротехнической лаборатории Токио ведется разработка потоковой машины «Сигма-1» для научных расчетов. Машина имеет древовидную структуру, содержащую 32 кластера[1], по восемь процессорных элементов в каждом. Эта структура требует соответствующей иерархической сети коммуникации. Каждый кластер может одновременно участвовать в решении 256 задач. Любой процессорный элемент с циклом продолжительностью 80 пс имеет собственную память программ и содержит пять устройств, работающих асинхронно в конвейерном режиме. Процессорные элементы взаимодействуют с помощью пакетной связи, а для кластера используется перекрестный переключатель. Производительность машины — 100 млн. оп/с.

Простейшими системами с управляемыми данными являются систолические матрицы, в которых фиксированная конфигурация самых элементарных процессорных элементов синхронно обрабатывает проходящий через систему поток данных. Название «систолическая» эта обработка получила из-за сходства между пульсациями крови в кровеносной системе и прохождением данных в матрице процессорных элементов. Идея систолической матрицы была предложена в 1978 г. профессором Г. Кунгом из университета Карнеги-Меллона (Питтсбург, США). Архитектура систолической матрицы обеспечивает высокую степень распараллеливания вычислительного процесса и его конвейерную организацию.

Систолические матрицы — высокоэффективное средство построения систем, ориентированных на определенный тип алгоритма. Настройка на определенный алгоритм может быть задана системой соединений между элементами матрицы. Если известно, что в процессе работы тип алгоритма меняться не будет, т. е. будет решаться одна и та же задача с различными исходными данными, то соединения можно задать конструктивным способом. Если же это предположение не имеет места, то необходим более гибкий способ задания соединений, так как их нужно будет изменять при переходе одной задачи к другой. Такие систолические матрицы называются программируемыми. Перепрограммирование в систолических матрицах происходит несложным путем, что выгодно отличает эту архитектуру от других архитектур, в которых имеются достаточно

[1] От английского cluster — гроздь, пучок. Кластер можно определить как ветвь древовидной структуры.

сложные и не слишком эффективные по затратам времени коммутаторы. Систолические матрицы относят к архитектуре типа ОКМД (одиночный поток команд, множественный поток данных). Программирование матриц приближает их к архитектуре типа МКМД (множественные потоки команд и данных).

С исследованиями по программируемым систолическим матрицам связан ряд проектов потоковых машин и процессоров сигналов. Под руководством Г. Кунга в первой половине 80-х годов (работа началась в 1981 г.) в университете Карнеги-Меллона (Питтсбург, США) был создан программируемый систолический однокристальный микропроцессор. Этот микропроцессор, сформированный на *n*-МОП кристалле, содержит память микропрограмм, регистровый накопитель, арифметическое логическое устройство, умножитель, три порта ввода и три порта вывода, соединенные независимыми шинами. В 1985 г. в университете Карнеги-Меллона была введена в действие 10-процессорная система Warp производительностью 100 млн. оп/с. В системе используются серийные микропроцессоры, работающие под управлением ведущего процессора. Основное назначение системы — выполнение операций типа быстрых преобразований Фурье, расчета операторов свертки, матричной арифметики.

Систолические матрицы особенно эффективны в качестве процессоров сигналов, при обработке которых типичны такие операции, как умножение одной матрицы на другую или матрицы на вектор. Всего во второй половине 80-х годов во всем мире на различных стадиях разработки находятся несколько десятков проектов потоковых ЭВМ и систолических матриц. Масштабы разработок связаны с перспективностью идеи обработки потоков данных, которая применима в универсальных сверхпроизводительных машинах для научных расчетов и в экспертных системах с мощными базами знаний, а также в специализированных ЭВМ (процессоры сигналов на систолических матрицах и т. д.).

В отличие от работ по созданию потоковых машин исследования в области редукционных машин более далеки от стадии практической реализации, хотя к середине 80-х годов было предложено около 20 отличающихся по архитектуре проектов. Как уже отмечалось, создание архитектуры редукционных машин базируется на попытках реализации идей функционального программирования.

Первые языки функционального программирования, такие, как ЛИСП[1], были созданы еще на рубеже 60-х годов, однако развитие они получили в первой половине 80-х годов, когда, с одной стороны, серьезно возросли трудности программирования[2], а с другой — стало ясно, что будущее вычислительных систем связано с параллельными структурами. Основные преимущества функционального стиля программирования определяются такими его чертами, как: 1) очень высокий уровень языка; 2) компактные и ясные нотации функциональных языков позволяют записать алгоритм меньшим числом строк кода; 3) наличие во многих программах, записанных на функциональном языке, явного и легкообнаруживаемого параллелизма; 4) более легкая верификация программ и отсутствие побочных эффектов. Именно эти черты соответствуют требованиям 80-х годов. Особенное значение имеет повышение производительности труда программиста: длина функциональной программы на порядок меньше, чем программы, записанной на ФОРТРАНе или другом традиционном языке высокого уровня.

Трудность, однако, состоит в том, что реализация функциональных языков на машинах с фон-неймановской архитектурой сопровождается катастрофическим падением производительности ЭВМ. Необходимы те или иные формы параллельных структур. Поточная архитектура в принципе го-

---

[1] ЛИСП сочетает черты функционального языка с некоторыми чертами традиционных языков. Полностью функциональным является язык чистый ЛИСП (pure LISP)

[2] В основе этих трудностей лежит несравненно более медленный рост производительности труда программистов по сравнению с ростом производительности машин.

раздо лучше приспособлена для реализации функциональных языков, чем фон-неймановская. С этим и связано то обстоятельство, что ряд потоковых ЭВМ ориентирован на использование функциональных языков. Разработаны и гибридные архитектуры, сочетающие свойства потоковых и редукционных машин.

Из проектов редукционных машин широкую известность получил проект профессора Г. Маго из университета штата Северная Каролина. Машина Маго предназначена для работы с языком функционального программирования, поскольку архитектура этой машины реализует функциональный язык, отпадает необходимость и в промежуточном языке программирования, и в операционной системе. Машина имеет бинарную древовидную структуру, число элементов которой в принципе не ограничено. В конечных узлах дерева — процессорные элементы, а в его корне — контроллер, осуществляющий общее управление системой. В течение всего вычислительного цикла происходят волнообразные передачи сигналов: контроллер посылает сигналы (исходные данные) процессорным элементам и получает ответные сигналы (результаты обработки данных). В первой половине 80-х годов была создана действующая модель многопроцессорной архитектуры ДОТ — расширенного варианта архитектуры машины Маго и разрабатывалась соответствующая трехуровневая система программирования.

Другой известный проект редукционной машины, получившей название «Алиса», разрабатывается в 80-е годы в Королевском колледже (Лондон). Дж. Дарлингтон создал для этой машины функциональный язык ХОУП. Однако «Алиса» не рассчитана на непосредственное выполнение программ, записанных на языке ХОУП, и использует ассемблерный язык. Структура машины предусматривает наличие быстрого централизованного переключателя, объединяющего пул процессорных элементов с большой матрицей памяти. Блоки данных и связанные с ними команды объединены в пакеты, одновременно обрабатываемые группой процессорных элементов. Количество процессорных элементов, которое позволяет использовать архитектура «Алисы», составляет несколько тысяч. В качестве процессорных элементов предусматривается применение либо транспьютеров (см. ниже), либо мощных микропроцессоров с памятью 1 Мбайт.

Использование мощных процессорных элементов связано с тем, что архитектура машины «Алиса» (наряду с архитектурой потоковой машины Манчестерского университета) избрана в качестве базовой для английского национального проекта ЭВМ пятого поколения. Аппаратная реализация «Алисы» в полном объеме (в Королевском колледже в начале 80-х годов был создан макет на 16 процессорных элементах) осуществляется английской фирмой ИКЛ, а консорциум во главе с фирмой «Плесси» создает на базе этой машины систему распознавания речи.

Рассмотренные выше проекты потоковых и редукционных машин далеко не исчерпывают всей совокупности выполняемых в мире исследований и разработок в этой области, тем не менее дают общее представление о направлении ведущихся в 80-е годы работ. В принципе оба направления не исключают друг друга. Например, в потоковых машинах возможно использование принципов функционального программирования. С этим обстоятельством связано наличие проектов, синтезирующих черты обоих типов архитектуры. К ним относится, например, проект многопроцессорной системы Редифлоу (университет штата Юта, США), в котором комбинируются свойства редукционных, потоковых и фон-неймановских машин.

Каждый процессорный элемент этой системы имеет список приоритетных задач (выполняемых по запросу других элементов) и список доступных данных. Идея организации вычислительного процесса состоит в том, что процессоры, не ожидая запросов, выполняют задачи в порядке их приоритета. Сеточная конфигурация процессорных элементов состоит из *x*-пьютеров (микропроцессоров, содержащих процессор, память и переключатель). Сеть образует поверхность с тремя логическими

параллельными слоями, причем устройство ввода-вывода может подключиться к любому узлу. Одна из целей проекта — оценка возможностей использования функционального языка и многопроцессорных систем. Применение функциональных языков предусмотрено в некоторых рассмотренных выше проектах потоковых машин, в том числе в машинах Арвинда, Денниса, Амамия и др.

Концепция РИСК-архитектуры и транспьютеры

Деление машин на фон-неймановские, поточные и редукционные — слишком «крупноблочное», чтобы отразить многие существенные черты архитектуры проектов машин 80-х годов. Один из альтернативных подходов к классификации связан с полнотой набора машинных команд. По этому признаку выделяются машины с сокращенным набором команд (архитектура РИСК) и полным набором (архитектура КИСК).

Идея архитектуры РИСК порождена, с одной стороны, развитием интеграции полупроводниковых схем, а с другой — стремлением реализовать минимальными аппаратными средствами машины с достаточно высокими вычислительными возможностями. Иными словами, эта идея оптимизации технологических возможностей и структурных решений, красной нитью проходящая через всю историю ЭВТ. В прошлом эта идея с наибольшим эффектом была реализована трижды: в фон-неймановской архитектуре, идеально отвечающей возможностям построения машин на электронных лампах, в идее мини-ЭВМ, оптимально приспособленной к переходу от дискретной к интегральной технологии, и в концепции микропроцессора, привязанной к технологии БИС. Идея архитектуры РИСК соответствует переходу к сверхбольшим интегральным схемам.

Название этой архитектуры было предложено Д. Паттерсоном из Калифорнийского университета в Беркли, а первые реализации связаны с тремя разработками. Первой РИСК-машиной стала мини-ЭВМ IBM-801, созданная в начале 80-х годов в ходе выполнения научно-исследовательского проекта, предложенного Дж. Коком. Это был маломощный компьютер, построенный на схемах с эмиттерными связями, с аппаратно реализованным управлением и небольшим количеством однотактных команд. В первой половине 80-х годов широкую известность в США получили два университетских проекта: универсальные микропроцессоры РИСК-1 и РИСК-2, созданные в Беркли под руководством Д. Паттерсона, и процессор МИПС, разработанный в Станфордском университете.

Отправной точкой идеи архитектуры РИСК послужило осознание того обстоятельства, что в программах обычных компьютеров сложные команды выполняются весьма редко. Среднестатистические данные показывают, что 80% всех команд используются в течение 20% машинного времени. Соответственно возникла идея разбить сложные операции на базовые элементарные команды, такие, как «записать», «сравнить», «сложение», «ветвление» и т. п., добившись тем самым роста производительности. В конечном итоге в ходе разработки упомянутых проектов сложились следующие принципы архитектуры РИСК: 1) упрощенный и фиксированный состав команд (все они имеют одинаковую длину и структуру); 2) аппаратная реализация управления вместо микропрограммного; 3) выполнение всех (или по крайней мере большинства) команд за один такт; 4) осуществление доступа к памяти только через команды загрузки в центральный процессор и записи в запоминающее устройство.

Архитектура РИСК обладает рядом преимуществ. Сокращенный набор команд позволяет строить эффективный компилятор. Интенсивное использование больших регистровых стеков уменьшает число обращений к памяти. Снижение аппаратурных затрат, естественно, улучшает такие характеристики, как надежность машины, ее экономичность, технологичность и т. п. Проведенные в первой половине 80-х годов сравнения машин с архитектурой РИСК с обычными компьютерами (в универси-

тете в Беркли и в университете Карнеги-Меллона) показали превосходство первых по производительности в 2—4 раза[1]. Проблематичной, однако, остается возможность непосредственного применения языков высокого уровня (одна из центральных задач проектирования ЭВМ пятого поколения) в машинах с РИСК-структурой.

В 1985 г. ЭВМ с архитектурой РИСК достигли уровня коммерческой реализации. В этом году корпорация IBM продолжала работы над 32-разрядным микропроцессором (вариант мини-ЭВМ IBM-801 на одном кристалле) производительностью 16,6 млн. оп/с. Фирма Hewlett-Packard (США) разработала семейство ЭВМ с РИСК-архитектурой. В то же время дальнейшее совершенствование технологии (рост степени интеграции полупроводниковых схем) позволило снизить жесткость требований, предъявляемых архитектурой РИСК «в чистом виде». В середине 80-х годов ряд коммерческих проектов нацелен на комбинацию РИСК-архитектуры с традиционной КИСК-архитектурой. В этой связи используется термин «РИСК-архитектура второго поколения». Примерами машин с такой архитектурой могут служить супер-мини-ЭВМ 32/100 (фирма «Ридж компьютерс», США), 90 и 98 («Пайрэмид технолоджи», США). В этих машинах реализованы возможности выполнения операций с плавающей запятой, используются более широкий набор команд (90—100), микропрограммирование и команды переменного формата.

Непосредственно с проектом машины пятого поколения связано применение архитектуры упоминавшегося выше микропроцессора МИПС Станфордского университета в качестве базовой для создания блоков бортового процессора, обрабатывающего сигналы в системе противоракетной обороны. Сверхбыстродействие, требуемое для такого процессора, реализуется в первую очередь за счет замены кремния арсенидом галлия. Основная задача — получить кристалл арсенида галлия с 10 тыс. вентилей, на базе которых реализуется 32-разрядный микропроцессор с уникальной тактовой частотой 200 МГц и производительностью 100 млн. оп/с. Из таких микропроцессоров предполагается создать сверхбыстродействующую систему для обработки сигналов.

При всей важности подобных проектов для развития ЭВТ крупнейшим, на наш взгляд, достижением, связанным с реализацией архитектуры РИСК, явилось изобретение транспьютера. По замыслу его создателей (специалистов английской фирмы «Инмос»), транспьютер проектировался как компонент многопроцессорных систем [3]. Именно эта ориентация и предопределила значение транспьютера для реализации концепций ЭВМ пятого поколения. Вычислительная система, состоящая из m-транспьютеров с быстродействием n оп/с каждый, практически ничего не теряет в суммарном быстродействии, которое, следовательно, составляет mn оп/с. Таким образом, открываются исключительно благоприятные перспективы для создания сверхвысокопроизводительных систем.

Транспьютер — коллективное произведение небольшой, поддерживаемой Британским научно-исследовательским советом фирмы, созданной в 1979 г. именно для разработки микропроцессора нового типа: с относительно емкой памятью, простым по архитектуре и быстродействующим центральным процессором и эффективными программируемыми средствами связи. Проектирование транспьютера было тесно связано с созданием языка ОККАМ, идеологом которого был Д. Мэт, разработавший совместно с И. Барроном (руководителем фирмы) основные принципы организации нового микропроцессора.

Язык ОККАМ относится к процедурным языкам высокого уровня. Он позволяет описывать систему, состоящую из набора транспьютеров, как ряд параллельных процессов, которые происходят в основном совмещенно и независимо друг от друга. Он может быть использован и для проектирования и для программирования (как сети транспьютеров, так и отдельного транспьютера).

[1] При сочетании архитектуры РИСК с конвейерным режимом обработки и наличием дополнительного процессора для выполнения операций с плавающей запятой.

Все аспекты языка ОККАМ реализованы в аппаратуре транспьютера, в результате чего отпадает необходимость в программных средствах для планирования процессоров и организации обмена данными. Проверку на практике язык ОККАМ прошел в ходе разработки транспьютера для системы автоматизированного проектирования.

Архитектура транспьютера может быть охарактеризована как РИСК-архитектура второго поколения. Используется микропрограммное управление. Набор команд выбран с целью обеспечения эффективной компиляции высокоуровневых языков ПАСКАЛЬ и СИ. Особое внимание уделено организации сквозных обменов при взаимодействии между транспьютерами и обеспечению совместимости транспьютеров с различной длиной слова. В целом архитектура ориентирована на системное использование транспьютеров.

В 1986 г. фирма «Инмос» начала серийный выпуск транспьютеров модели Т 414. Эта модель, содержащая около 250 тыс. компонентов на кристалле кремния площадью 76 мм^2, имеет оперативную память емкостью 2 кбайта, постоянное ЗУ и четыре асинхронных канала связи с суммарной пропускной способностью 5 Мбайт/с. Модель Т 414 оперирует 32-разрядными словами с быстродействием 10 млн. оп/с. Следующая модель (Т 424), также выпущенная в 1986 г., имеет оперативную память 4 кбайта. Дальнейшие исследования, связанные с разработкой транспьютера второго поколения, предусматривают дополнительное размещение на одном кристалле двух процессоров для умножения чисел с плавающей запятой и для управления передачей данных.

Системное применение транспьютеров, как уже отмечалось, весьма перспективно. По состоянию на середину 80-х годов оно ограничивается единичными системами. Для решения задач машинной графики и демонстрации системных возможностей транспьютеров фирма «Инмос» построила систему из десяти Т 414, имеющую производительность 100 млн. оп/с. В 1985 г. английская фирма «Мейко» разработала на базе транспьютеров систему, получившую название «Вычислительная поверхность», производительностью 750 млн. оп/с. Специализированная система из 100 транспьютеров, сопряженная с подсистемой дисковых накопителей, используется в Великобритании (г. Кобем, графство Суррей) в автоматической системе идентификаций отпечатков пальцев. Номинальное быстродействие системы — до 5 млрд. оп/с. С проектами машин пятого поколения связано применение транспьютеров в упоминавшейся ранее редукционной машине «Алиса».

Топология связей между процессорами

Топология связей между процессорными элементами имеет важное значение для характеристики архитектуры многопроцессорных машин. Для проектов 80-х годов характерны исключительно разнообразные типы связей. В частности, сети связей имеют вид дерева, кольца, звезды, тора, треугольника, гиперкуба, обобщенного гиперкуба, двумерной и трехмерной решетки и др.

К числу наиболее распространенных относится структура в виде двоичного дерева, характерная для ряда алгоритмов управления и вычислений. Эти структуры характеризуются минимальным числом соединений на один процессор.

Широко распространена одномаршрутная и многоуровневая сеть пакетных переключений, например «Омега». Одновременно исследуются и разрабатываются многоуровневые многомаршрутные сети.

Внимание к топологии связей нашло отражение в названиях некоторых проектов. Известность получил проект «Космический куб», названный так его автором Ч. Сейцем из Калифорнийского технологического института (г. Пасадена). Процессорные элементы этой системы рассматриваются в проекте как узлы сети связи, называемой двоичным шестимерным кубом (частный случай гиперкуба), где каждый узел соединен двунаправленными асинхронными каналами с соседними узлами. Примененная топология связей позво-

ляет систематически наращивать число процессорных элементов. «Космический куб» представляет собой слабосвязанную мультипроцессорную систему, в которой каждый процессорный элемент имеет локальную память. Экспериментальный образец «Космического куба» на 64 процессорных элементах был введен в эксплуатацию в октябре 1983 г. Следующие этапы проекта — создание систем из 128 и 1024 узлов.

«Космический куб» — один из экспериментальных проектов 80-х годов, перешедший в стадию серийных коммерческих разработок. Их осуществляет, естественно, не институт, а промышленная фирма (INTEL), анонсировавшая в 1985 г. систему iPSC. Система представлена тремя моделями, объединяющими 32, 64 и 128 микропроцессоров (типа Intel 80 286/80 287 с тактовой частотой 8 МГц и локальной памятью 0,5 Мбайта в первоначальном варианте и 4 Мбайта в следующем). В качестве базового языка используется Коммон ЛИСП [4].

Топология связей «Космического куба» используется в качестве базовой в нескольких проектах суперсистем, в том числе в разрабатываемом с 1984 г. в Принстонском университете компьютере с максимальным быстродействием 60 млрд. оп/с. Разработка ведется под руководством Д. Нозенчука и финансируется НАСА. Архитектура соединения узлов в этом компьютере — упрощенный вариант архитектуры «Космического куба». Основная структурная единица компьютера — узел, содержащий 24 мощных 32-разрядных микропроцессора и до 4 Гбайт памяти. Быстродействие такого узла — около 400 млн. оп/с, что позволяет достичь заданной производительности системы за счет объединения 128 узлов.

Связь между узлами организуется с помощью регистровых блоков в матричном коммутаторе. Скорость обмена данными между узлами ограничивается пропускной способностью волоконно-оптического канала связи (500 Мбит/с). Матричный коммутатор представляет собой сетевую структуру с топологией Бенеша, т. е. неблокирующую сеть с динамической реконфигурацией, позволяющую одновременно осуществлять соединения любых двух входов и выходов. Подобная сеть дает возможность каждому процессору запросить любые данные и получить их через семь тактов.

Хотя задание НАСА заключалось в создании специализированной системы (для моделирования возникновения турбулентных режимов в потоках жидкости), очевидно, что проектируемый компьютер способен выполнять широкий круг задач, связанных с большими объемами вычислений, допускающих применение векторных методов и распараллеливания алгоритмов. К таким задачам, в частности, относится значительная часть задач моделирования распознавания образов и машинной графики.

Внимание к топологии связей отразилось в названии проекта «Машина соединений» (Connection Machine), разработанного в Массачусетсском технологическом институте и реализуемого фирмой «Думающие машины» (Thinking Machines).Основные идеи, лежащие в основе проекта, принадлежат Д. Хиллису, до 1983 г. работавшему в МТИ, а затем возглавившему созданную в том же году фирму. Первым изделием фирмы и стала «машина соединений» (1986 г.).

Метод соединений, примененный в машине, — *n*-кубическая сеть. В первой модели «машины соединений» (предполагается разработка семейства машин) используется 65 536 одноразрядных микропроцессоров с памятью емкостью 4096 бит, т. е. суммарная память системы составляет 32 Мбайта. Технология сверхбольших интегральных схем позволила разместить на одном кристалле 16 одноразрядных микропроцессоров (по 10 тыс. вентилей в каждом) и один узел сети микропроцессорной связи, соединяющей эти микропроцессоры. Через сеть связи («коммутатор — маршрутизатор») любой микропроцессор может посылать сообщение любому другому из 65 тыс. микропроцессоров. Соответствующие каналы связи устанавливаются программным способом и могут изменяться в процессе выполнения программы: при решении одной задачи конфигурация связей между микропроцессорами может

иметь вид решетки, при решении другой — кольцевую схему и т. д. Общая пропускная способность сети 3 млрд. бит/с.

В «машине соединений» реализован параллелизм на уровне данных, т. е. одна и та же задача выполняется на всех процессорах, каждый из которых обрабатывает свой элемент данных. Ясно, что этот способ эффективен при решении задач с большими объемами данных. Производительность машины — от 1 млрд. оп/с при решении вычислительных задач общего характера до 6—7 млрд. оп/с при выполнении специализированных задач, таких, как поиск документов, моделирование потоков жидкости и т. п. Одно из основных применений машины — проектирование сверхбольших интегральных схем. При их разработке осуществляется моделирование десятков и сотен тысяч транзисторов, размещенных на одном кристалле. В этом случае в машину поступают команды, по которым каждый процессор осуществляет пошаговое моделирование «своего» транзистора. Иными словами, каждый процессор выполняет одну и ту же последовательность команд, но для собственных (уникальных) данных. Этим способом вычисляются такие параметры, как напряжение, ток и проводимость для тысяч транзисторов одновременно. Если транзисторов на схеме больше 65 тыс., аппаратура машины переводится в виртуальный режим, когда один физический процессор имитирует несколько виртуальных[1]. Система позволяет формировать до 1 млн. виртуальных процессоров.

Говоря о «машине соединений» с производительностью минимум 1 млрд. оп/с, нельзя не упомянуть, что вся машина, сконструированная по форме, близкой к кубу (142 и 142×157,5 см), имеет массу 1180 кг и скомпонована на 153 многослойных печатных платах (в том числе 128 процессорных плат размерами 38×45,3 см, на каждой из которых 32 сверхбольшие интегральные схемы). Стоимость этой машины, содержащей около миллиарда транзисторов, — 3 млн. долл.

Примененная в рассмотренных выше проектах топология связей типа гиперкуб представляет собой частный случай метода соединений «обобщенный гиперкуб», обладающего высокой устойчивостью к отказам. Обобщенная гиперкубическая архитектура включает в себя в качестве частных решений обычный гиперкуб, кольцо, решетки с попарным соединением соседних узлов в многомерном пространстве.

Альтернативным типом этих решений является древовидная структура, которая используется, в частности, в упоминавшемся выше проекте машины Маго. Особенность этой структуры — минимум числа соединений и наличие только одного пути между каждой парой узлов. Частными случаями древовидной топологии служат звездообразная, радиальная и некоторые другие типы соединений. Разумеется, возможны комбинации древовидной структуры с другими типами связей.

К подобным относится система, разрабатываемая по проекту HOH-BOH-1 в Колумбийском университете под руководством Д. Шоу. Конечная цель проекта — объединение около 6 тыс. процессоров с достижением максимальной производительности 25 млрд. оп/с. По состоянию на 1985 г. создана экспериментальная система с древовидной структурой, состоящей из одного 32-разрядного микропроцессора, координирующего работу 63 процессорных элементов (4-разрядных микропроцессоров с постоянной памятью емкостью 64 байта). Экспериментальная система представляет собой одну ветвь будущего дерева. Каждая такая ветвь связана с главной ЭВМ, размещенной в корне дерева. Кроме того, предусмотрена связь находящихся в основании ветвей 32-разрядных микропроцессоров между собой. В ходе вычислений эти микропроцессоры, работающие в режиме обработки множественных потоков команд и данных, распределяют потоки команд к 4-разрядным микропроцессорам, выполняющим одну операцию над множеством данных.

Древовидные соединения, также

[1] Разумеется, виртуальный режим используется не только при моделировании, но и во всех случаях, когда элементов данных больше 65 тыс. Если меньше — излишние микропроцессоры отключаются.

как и соединение типа обобщенного гиперкуба, относятся к классу одноуровневых. В более сложных многоуровневых сетях осуществляется эффективное соединение каждого процессора с каждым модулем памяти. В таких сетях используются два или более уровней промежуточных переключающих узлов. Машины с такой топологией связей разрабатываются в ряде проектов 80-х годов: «Баттерфлай», «Ультракомпьютер» и др.

Система «Баттерфлай» («Бабочка») разработана американской фирмой BBN для решения задач искусственного интеллекта. Создание этой системы является составной частью проекта работ министерства обороны в области ЭВМ пятого поколения. Машина предназначена для таких целей, как проектирование автономного космического аппарата, разработка программы управления наземно-воздушным боем и др. Первый вариант этой системы содержал 10 процессоров, в середине 80-х годов были разработаны системы из 128 и 256 процессоров (производительностью около 180 млн. оп/с). Подсчитано, что суммарная производительность системы линейно возрастает по мере увеличения числа процессорных элементов (на 0,5 млн. оп/с с добавлением каждого нового процессора).

Проект «Ультракомпьютер» осуществляется в Нью-Йоркском университете под руководством А. Готлиба. Конечная цель проекта — создание к 1990 г. системы из 4096 процессоров производительностью 10 млрд. оп/с. По архитектуре это сильносвязанная система с глобальной памятью и многоуровневой соединительной сетью типа «Омега». В середине 80-х годов разработан образец из восьми процессоров, в качестве которых использованы серийные микропроцессоры (68 010 фирмы Motorola).

Идеи проекта «Ультракомпьютер» легли в основу крупного исследовательского проекта корпорации IBM, целью которого является создание RP3-экспериментального компьютера для научных расчетов. Во второй половине 80-х годов планируется завершение работ над вариантом, содержащим 64 процессора, и в случае успешных испытаний — создание системы из 512 процессоров.

Каждый из процессоров представляет собой микропроцессор с архитектурой РИСК и памятью 4 Мбайта. Отличие от архитектуры проекта «Ультракомпьютер» заключается в том, что память системы в зависимости от характера решаемой задачи и целей пользователя может рассматриваться либо как глобальная, либо как полностью локальная [5].

В СССР в середине 80-х годов возникло несколько проектов ЭВМ, архитектура которых допускает волновое перемещение информации по однородному процессорному полю. Такой подход (как и те подходы, которые использованы в машине Д. Хэллиса или проекте НОН-ВОН-1) основан на совершенно новой идеологии программирования. Необходимы средства описания не линейной последовательности команд, образующих программы в ЭВМ традиционной архитектуры, а некоторой совокупности асинхронных параллельно протекающих процессов. Теория такого программирования пока еще только зарождается, что и приводит к использованию «волновой идеологии» при решении тех задач, для которых удалось найти модель описания в виде движения некоторых «фронтов активности» по процессорным элементам. К наиболее известным классам таких задач относятся те, которые по своей сути сводятся к задачам на графах и сетях [6]. Построение же универсальных ЭВМ на уровне «волновой идеологии» пока не удается.

Поэтому наиболее интересны проекты, в которых подобный подход используется при построении спецпроцессоров баз знаний. Если для представления знаний используются семантические сети, то удается получить эффективные процедуры для поиска вывода на знаниях, обобщения знаний, внесения изменений в базу знаний. Такой спецпроцессор разрабатывается в рамках проекта ПАМИР специалистами из СССР и ЧССР [7]. Несколько иной подход используется в проекте ЛИВС, в выполнении которого участвуют специалисты из ВНР и СССР [8]. Оба эти проекта выполняются в рамках про-

граммы совместных работ по созданию ЭВМ новых поколений, объединяющей усилия специалистов из социалистических стран.

Проекты ЭВМ новых поколений показывают, что их новой парадигмой является переход к асинхронности и параллельности. Такой переход требует развития специальной теории решения задач, опирающейся на асинхронно протекающие взаимосвязанные процессы. Другими словами, нужно создавать новую теорию программирования, концепции которой резко отличны от тех, которыми руководствуется программирование для ныне существующих ЭВМ.

Что же касается аппаратных решений, то они явно обогнали математическое обеспечение будущих ЭВМ. В области реализации ЭВМ многое уже решено. Выявились две основные тенденции построения будущих ЭВМ: объединение в многопроцессорную систему однотипных процессорных элементов и создание многопроцессорной системы из специализированных устройств. Выбор того или иного пути во многом зависит от оценки того влияния, которое оказывает на степень возможного параллелизма функциональная сложность процессорных элементов.

В соответствии с классификацией Дж. Шварца, предложенной в 1983 г., все проекты ЭВМ с параллельной структурой, выполняющиеся в 80-е годы, подразделяются на две группы: с использованием крупных и минимальных (по аппаратуре) процессорных элементов.

Минимальные по аппаратуре и архитектуре процессорные элементы (типа однобайтовых или даже одноразрядных микропроцессоров) обеспечивают параллелизм на командном уровне (конвейерная, векторная обработка) и даже на разрядном уровне (например, упоминавшаяся выше «машина соединений»). В таких системах процессорные элементы или уровни конвейеров требуют синхронизации на субнаносекундном уровне.

Максимальными возможностями повышения производительности обладают системы из больших процессоров, в каждом из которых, в частности, могут содержаться минимальные процессорные элементы, обеспечивающие конвейерную обработку. Большие процессоры обеспечивают универсальность системы и отвечают современным представлениям, которые предполагают разбиение проблемы на отдельные задачи, выполняемые независимо друг от друга. В проектах 80-х годов большие процессоры используются в машинах Денниса, Арвинда и др.

Промежуточное положение между минимальными и максимальными по своим возможностям процессорами занимают процессорные элементы «средней мощности», такие, например, как в проектах «Ультракомпьютер» и RP3. С их помощью обеспечивается параллелизм на уровне выполнения последовательных итераций. При этом используются специальные параллельные компиляторы, позволяющие менять режим вычислений. Для систем этого типа требуется синхронизация на микросекундном уровне.

Особенность зависимости степени параллелизма от функциональной сложности процессорных элементов состоит в том, что ситуация носит динамический характер: развитие технологии позволяет при тех же затратах создавать все более сложные структуры. Это нередко отражается на ходе выполнения одного и того же проекта, когда вместо примененных в первых экспериментальных вариантах процессорных элементов на дальнейших этапах проекта используются более сложные. Быстрое поступательное развитие полупроводниковой технологии предопределяет возможность применения все более высоких по производительности микропроцессоров в качестве компонентов многопроцессорных вычислительных систем.

Чем вызван переход к интеллектуальным ЭВМ новых поколений?

Огромные усилия, затрачиваемые во всех развитых странах на развитие работ в области искусственного интеллекта и создание ЭВМ новых поколений, оправданы теми причинами, которые лежат в основе этих устремлений.

Эти причины можно разделить на два типа: непосредственные и опосредованные. Непосредственные причины определяются самой логикой развития науки и вычислительной техники. Причины второго типа тесно связаны с крупномасштабными социальными целями, определяемыми нуждами научно-технического прогресса.

Одна из основных непосредственных побудительных причин обусловлена проявлявшимся на протяжении всей истории ЭВТ недостатком вычислительных мощностей для решения сложных задач. Сказать, что такая ситуация будет всегда, было бы слишком обязывающим. Тем не менее ясно, что быстродействие порядка тысячи операций в секунду, на которое большие надежды возлагал Ален Тьюринг, когда создавались первые ЭВМ, оказалось столь же недостаточным, как и производительность 1 млрд. оп/с, достигнутая в середине 80-х годов. В самом общем виде стремление повысить быстродействие обусловлено ростом размерности решаемых задач и, возможно, нашим неумением решать их более экономными средствами. Задача увеличения быстродействия закономерно привела к попыткам реализации различного рода параллельных структур, обещавших качественное повышение производительности. Сама же реализация параллельных структур стала практически возможной благодаря развитию полупроводниковой технологии.

Другая побудительная причина — растущие трудности в области программирования. Эти трудности во многом снимались в случае реализации аппаратными средствами языков высокого уровня и создания систем, преобразующих естественный язык во внутренний язык машины. Все это было прекрасно известно и раньше, но только развитие электронной технологии и разработки в области алгоритмических языков сделали актуальными постановку соответствующих задач.

Третья побудительная причина связана с общей тенденцией применения вычислительной техники для решения задач логического характера в самых различных областях — от поиска информации и автоматизированного обучения до управления объектами в реальных условиях изменяющейся окружающей среды. Основой для разработок в этих направлениях послужили предшествующие исследования в различных областях искусственного интеллекта, а средствами технической реализации — сверхбольшие интегральные схемы, мощные системы магнитной и оптической памяти, волоконная оптика и т. д.

Таковы непосредственные побудительные причины, связанные с выдвижением проектов машин пятого поколения. А поскольку эти проекты, начиная с Японии, стали выдвигаться на национальном уровне, к «внутренним» побудительным причинам следует добавить «внешнюю» — стремление того или иного государства захватить лидерство (или, по крайней мере, не отстать от лидера) в области обработки информации. Однако было бы неверно ограничиваться только этими причинами и упускать из виду более крупные социальные цели, возможность достижения которых опирается на общие закономерности НТП.

К числу таких закономерностей относится стадийность процесса автоматизации, заключающаяся в последовательной смене трех фаз автоматизации: жесткой, гибкой, на основе искусственного интеллекта [9].

С помощью средств жесткой автоматизации (типа автоматических станочных линий) теоретически можно было бы автоматизировать основную часть современной производственной сферы при одном (невыполнимом) условии: почти неизменном выпуске одних и тех же изделий с неизменными параметрами. Утопичность подобной идеи очевидна. Переналадка средств гибкой автоматизации (в отличие от жесткой) требует только смены программы. Так работают станки с ЧПУ, роботы, современные ЭВМ. С помощью средств гибкой автоматизации можно было бы комплексно автоматизировать значительное большинство видов работ, выполняемых в производственной сфере. Тем не менее за пределами возможностей автоматизации остались бы многие процессы, начиная от сборки наиболее сложных конструкций и кончая приня-

тием решений по тем или иным производственным вопросам. Если же мы хотим освободить человека от монотонной и утомительной работы на поточных линиях, сложных погрузочно-разгрузочных и других видах работы, недоступных современным автоматам, необходимо применение устройств с искусственным интеллектом. Только с их помощью возможна комплексная автоматизация производственной сферы, а также тех областей непроизводственной сферы, где это целесообразно. Именно в этом заключается историческая роль исследований по искусственному интеллекту, их значение для цивилизации.

Литература

1. Деннис Дж. Б., Россин Дж. Б., Линденман Дж. П. Схемы потока данных // Теория программирования. — Новосибирск, 1972. — Ч. 2. — С. 7—43.

2. Худик Я. Архитектура потоковых вычислительных машин // Алгоритмы, математическое обеспечение и архитектура многопроцессорных вычислительных систем. — М.: Наука, 1982. — С. 75—103.

3. Баррон Н., Кэвил П., Мэй Д. Транспьютер с быстродействием 5 млн. оп/с и более // Электроника. — 1983. — № 23. — С. 26—35.

4. Мануэль Т. Перспективы создания в США компьютеров пятого поколения // Электроника. — 1985. — № 26. — С. 3—5.

5. Нэйгел Т. Проект многопроцессорного компьютера с параллельной архитектурой // Электроника. — 1985. — № 18. — С. 6—7.

6. Сапатый П. С. Активное информационное поле как модель структурного решения задач на графах и сетях // Известия АН СССР. Сер. «Техническая кибернетика». — 1984. — № 5. — С. 26—36.

7. Вагин В. Н., Захаров В. Н., Поспелов Д. А. и др. Проект ПАМИР // Известия АН СССР. Сер. «Техническая кибернетика». — 1988. — № 2. — С. 181—170.

8. Гергей Т., Поспелов Д. А. Проект ЛИВС — логическая информационно-вычислительная система // Известия АН СССР. Сер. «Техническая кибернетика». — 1986. — № 5. — С. 128—138.

9. Апокин И. А. Кибернетика и научно-технический прогресс. — М.: Наука, 1982.

(Продолжение)

имеют одинаковую передаточную функцию. Элементы могут работать либо непрерывно, либо периодически в зависимости от заложенной функции. Каждый вход элемента имеет соответствующий вес, который может меняться по величине и знаку в процессе вычислений. Поэтому вес входа определяет силу связи.

Обучение нейросети осуществляется на ряде примеров. На вход подается заданная последовательность, а выходные данные сравниваются с правильными ответами. После каждого расчета вносятся поправки в коэффициенты и весовые функции. Так продолжается до тех пор, пока ошибки не достигнут приемлемых значений. После этого веса фиксируются, и сеть готова к работе. В некоторых случаях процесс обучения на этом полностью прекращается (например, при восприятии рукописного текста), в других — продолжается, если меняются условия, например, при расчете орбиты спутника Земли и расхода топлива.

В настоящее время уже находится в эксплуатации 13 различных нейрокомпьютеров. Все они имеют одинаковую архитектуру соединений.

Для изготовления нейрокомпьютеров применяются различные физические элементы: электронные, электронно-оптические или оптические.

Наиболее важна архитектура сети. Все разработанные к настоящему времени нейрокомпьютеры выполнены в виде сопроцессора, подсоединяемого к серийному компьютеру через шину передачи данных или через стандартные периферийные связи PC, Ethernet, DRV-11. В качестве сопроцессоров нейрокомпьютеры представляют собой периферийные устройства, такие же, как печатающее устройство или дисковод. Эти особенности архитектуры объясняются тем, что серийные ЭВМ имеют развитые средства ввода-вывода данных, а нейрокомпьютеры их не имеют.

Большинство нейрокомпьютеров предназначено для решения конкретной задачи. Нейрокомпьютеры, разработанные фирмами IBM, Texas Instruments и TRW, позволяют решать несколько классов задач; нейрокомпьютеры, разработанные фирмами HNC и SAIC, универсальны.

Процесс поведения или адаптации нейрокомпьютера может задаваться с помощью языка высокого уровня. Поскольку эта область совсем новая, используют только четыре языка: P3, Panspec, Anspec и Axon.

Появляется опыт использования нейросети в самых различных областях. Фирма Behavioristics Inc. продемонстрировала работу нейросети в банковском деле для определения степени надежности клиентов, берущих ссуды. Нейросеть показала лучшие результаты, чем ЭВМ, оперирующая на экспертной системе и статистической модели.

Профессор Калифорнийского университета Дж. Елман показал, что нейросети могут быть использованы для распознавания человеческой речи и преобразования ее в текст. При этом допускается слитная речь (без раздельного произношения слов). Эффективная система распознавания речи была разработана профессором Т. Коненом (Хельсинки) по контракту с японской фирмой Asahi Chemical Co. Профессор Конен показал также, что с помощью нейросети возможно распознавание образа лица человека и идентификация его с образами (до 500), хранящимися в памяти. Нейросеть позволяет распознавать рисунок пальца человека с 95% точностью при различных позициях, масштабе и даже небольших повреждениях. Перспективно использование нейросетей в робототехнике для управления движением.

IEEE Spectrum, 1988, 25 — № 3. — P. 36—41.

ЭВМ не смогут достичь человеческого уровня при решении задач до тех пор, пока программы не станут похожими на людей в двух отношениях: способности накапливать «жизненный опыт» и способности обучаться, общаясь друг с другом.

С. В. СЕМЕНИХИН

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СУПЕР-ЭВМ

Программное обеспечение — это комплекс программных средств, позволяющий в удобной форме использовать возможности вычислительной машины. Любая вычислительная система на уровне электронных схем (аппаратуры) ведет отработку данных, представленных в виде битов информации. В памяти машины расположены коды программ — последовательность операций и данные для этих операций — операнды. Центральный процессор, обращается к оперативной памяти, получает очередную операцию и операнды и выполняет необходимые действия, соответствующие конкретной команде.

Понятно, что человек (пользователь) не в состоянии на уровне битов сформировать задание (программу и данные) для ЭВМ, разместить это задание в памяти и в понятном для него виде получить результат. Необходимы более естественные и удобные средства и для указания последовательности действий на аппаратном уровне. Это и является основной функцией программного обеспечения. Прежде чем перейти к обсуждению компонентов программного обеспечения, необходимо рассмотреть основные аппаратные возможности, типичные для любой машины, где следует выделить центральные (ЦУ) и внешние устройства (ВУ).

К центральным устройствам относятся центральный процессор (ЦП), процессор ввода-вывода (ПВВ), оперативная память (ОП). Оперативная память состоит из машинных слов, каждое из которых имеет свой номер (адрес) и может быть считано в центральный процессор. В словах оперативной памяти размещаются коды программ (операции) и данные. Основная функция центрального процессора — исполнение операций (команд) некоторой программы. Процессор ввода-вывода по управляющей команде автономно осуществляет обмен информацией между заданной областью ОП и внешним устройством. Параллельно, в конкретный момент времени, ПВВ может управлять обменом информацией с несколькими внешними устройствами. Внешние устройства можно разбить на два класса: устройства для хранения больших объемов информации — магнитные диски и терминальное оборудование. К терминальному оборудованию относятся устройства типа дисплеев и устройства для ввода информации, например АЦПУ.

Основными компонентами программного обеспечения являются операционная система, языки программирования, диалоговые (интерактивные) средства работы с терминалом, прикладные пакеты для решения конкретных информационных или научно-технических задач. Все эти компоненты можно расположить в некоторой иерархической последовательности по отношению к аппаратным средствам: пользователь, прикладные пакеты, диалоговые средства, язык программирования, транслятор, операционная система, аппаратура.

Таким образом, составленная иерархия несколько условна и не означает, например, что пользователь применяет только средства, представленные прикладными пакетами. Приведенная иерархия характеризует степень «отдаленности» компонентов от аппаратуры и наиболее тесные границы взаимодействия.

По отношению к компонентам программного обеспечения существуют различные классы пользователей. Некоторые работают только со средствами, предоставляемыми прикладными пакетами, другие создают новые прикладные программы, третьи разрабатывают новые диалоговые средства или языки программирования, но это не значит, что при создании, например, новых языков нельзя использовать существу-

ющие диалоговые средства. Более того, сама ОС создается на некотором базовом языке программирования[1]. Определяющими, наиболее важными компонентами являются операционная система и система программирования (язык и транслятор). К основной функции ОС относится поддержка (реализация) необходимых возможностей языка программирования, управление работой центральных устройств и организация работы по обмену информацией с внешними устройствами.

Язык программирования позволяет представить необходимый алгоритм в некотором текстовом виде.

Для любого языка программирования существует специальная программа-транслятор. Текст программы на языке представляет собой исходную информацию для работы транслятора. По этому тексту транслятор формирует код программы (генерация кода).

Для чтения текста программы, например, с терминала или магнитного диска и записи полученного кода на магнитный диск транслятора используют средства, предоставляемые ОС. Основной функцией диалоговых систем являются разбор приказов, набираемых на терминале, или анализ изменений информации на экране, и запуск необходимой реакции (соответствующих программ). Для чтения информации с терминала и записи на него результатов исполнения приказов диалоговые системы используют или возможности языка или средства, предоставляемые ОС (языковые средства в конечном итоге преобразуются в соответствующее обращение к ОС).

Языки программирования. В рамках вычислительной системы действует несколько языков программирования и соответствующих им трансляторов. Некоторые из них используются для создания прикладных программ, некоторые — для создания трансляторов и ОС. Как правило, в большинстве систем существует язык ассемблер, позволяющий в символьном виде использовать операции — команды ЦП — и в таком виде предоставлять необходимый алгоритм. Используются языки и с более широкими, лаконичными и удобными возможностями — языки высокого уровня (Фортран, Алгол, Паскаль и др.).

На ассемблерах создаются, как правило, ОС и трансляторы, на языках более высокого уровня — прикладные системы. Однако следует отметить принципиальную разницу между ассемблером и языками высокого уровня. Кроме различного рода изобразительных средств, принципиальное отличие заключается в организации защиты данных. При использовании ассемблеров ошибки в программах могут привести к искажению информации. В языках высокого уровня такое недопустимо. Очевидно, хорошим качеством системы можно назвать наличие базового языка высокого уровня, на котором можно создавать и ОС, и трансляторы, и прикладные программы.

Определяющими понятиями языков программирования являются «переменная» и «процедура». Им ставится в соответствие некоторое имя. Такое соответствие устанавливается в тексте программы с помощью соответствующей конструкции языка, называемой «описание».

Пример.

.....
.....
.....

Переменная A, B, C, ... K ...;
Программа, в которой введены переменные с именами A, B, C... Подчеркнутые слова являются служебными для транслятора

Используя имя переменной (идентификатор) в тексте программы, можно изменять значение переменной.

Пример.

...A: =3.14; ...C:= A; ...K: = A+C*Д; ...

Приведенные конструкции транслируются в определенную последовательность команд и исполняются ЦП. Имени переменной при исполнении операции соответствует определенный адрес слова оперативной памяти.

При описании процедуры, кроме ее имени, указываются текст процедуры, определяющий некоторую последова-

[1] В этой статье не рассматривается вопрос о первоначальном создании операционной системы для новой машины.

тельность действий, которая при трансляции преобразуется в соответствующую последовательность команд. Каждая процедура может быть вызвана для исполнения в какой-то точке программы. Транслятор, преобразуя такой вызов процедуры (оператор процедуры), сформирует необходимую последовательность операций. В результате исполнения этих операций ЦП перейдет на исполнение первой команды соответствующей процедуры. При описании процедуры могут быть объявлены и так называемые формальные параметры. Для них при исполнении, как и для переменных, отводятся соответствующие слова в памяти машины. Значения для этих «параметров — переменных» устанавливаются в момент вызова процедуры, т. е. в этот момент формальные параметры принимают фактические значения. Если в тексте процедуры используются идентификаторы формальных параметров, то при исполнении процедуры в вычислениях будут участвовать фактические значения этих параметров

Описания переменных и процедур могут встречаться в начале определенных языковых конструкций, например, в начале программы или процедуры.

Следует отметить, что разница между программой и процедурой условна. Программа — это такая же процедура, только ее имя — это так называемое внешнее имя, которое определяет, где находится код программы для ее запуска. Запуск таких программ-процедур осуществляется с помощью средств, предоставляемых СС.

Процедура обладает еще одним очень важным свойством — контекстом исполнения. Контекст исполнения определяет имена всех объектов (переменных, процедур...), которые доступны для конкретной процедуры.

Например, если в программе описаны переменные А, В, С и процедуры П1 и П2, в тексте которых, в свою очередь, описаны соответственно переменные А1, В1, С1 и А2, В2, С2, то контекст процедуры П1 составляют переменные А, В, С, А1, В1, С1, а контекст процедуры П2 — А, В, С, А2, В2, С2.

Пример.

.....

.....

.....

Переменная А, В, С,... К...; описания переменных

.....

.....

.....

Процедура ПI=ПРОЦ (Х, У) описание текста процедуры с именем ПI

начало

●

К:=(Х+С):У

конец;

.....

.....

П1 (А, 1024)

Вызов процедуры ПI. В результате работы ПI значение переменной К станет (А+С):1024

В приведенных выше примерах не было ничего сказано о типах различных значений (объектов), над которыми можно производить те или иные операции. В большинстве языков при описании переменных определяется и ее тип. Такие языки являются статическими по типам. В динамических языках в различные моменты исполнения могут быть различные по типам значения переменной. Не нарушая общего хода дальнейших рассуждений, можно считать, что с некоторым идентификатором в момент исполнения операции связано значение, определяющее некоторый объект.

К основным объектам, которые вводятся в языки, относятся: целые и вещественные числа, логические значения, массивы, процедуры, файлы, справочники и др.

Пример.

```
. . . . . .
. . . . . .
X:= 3.14;
. . . . . .
. . . . . .
M :=МАССИВ [100]
. . . . . .
. . . . . .
НАЧАЛО
ПЕРЕМЕННАЯ М1...;
МАССИВ М2 50, М3 1000, ...;
. . . . . .
. . . . . .
Ф :=ФАЙЛ МД;
СПР:= СПРАВОЧНИК;
ПЕЧ := ФАЙЛ АЦПУ.
```

В приведенном примере 3.14 — изображение вещественного числа, переменной М, присваивается значение, которое является представителем массива из 100 элементов, имя М2 объявлено как массив из 50 элементов, значением переменной Ф станет указатель на ФАЙЛ (область памяти, где можно хранить данные), а переменной СПР — справочник, где можно хранить имена (внешние) созданных файлов и справочников.

Конструкции типа МАССИВ 100 или ФАЙЛ МД, используемые в правой части оператора присвоения (:=), называются генераторами объектов, конструкции типа МАССИВ М2 50, М3 1000 — объявлением. Не вдаваясь в конкретный анализ подобных конструкций, можно считать, что в языках есть средства создания объекта и возможность обращения к нему по имени (идентификатору).

Для работы с объектами различных типов разрешен только определенный набор операций. Например, числа можно складывать, умножать и т. п., массивы — индексировать, т. е. получать необходимый элемент массива. Например: 1 := 5; K := M1, здесь значением переменной К станет значение пятого элемента массива М. Над файлами разрешены операции записи и чтения, т. е. содержимое некоторого массива можно записать на файл и т. п. Например, ЗАП (М, ПЕЧ) — запись содержимого массива М на ФАЙЛ АЦПУ, т. е. печать.

Следует отметить, что объекты типа «файл» и «справочник» являются внешними объектами и часто реализуются в рамках так называемых языков управления заданиями или в приказах диалога, а в программу такие объекты передаются через параметры или специальные средства запуска программ.

Таким образом, используя средства конкретного языка программирования, можно создавать различные программы для решения необходимых задач, транслировать программы, полученный код, как файл, размещать на внешнем носителе и затем обращаться к нему по внешнему имени.

Пример.

Начало // транслятор (// ТЕКСТ, // КОД) конец

В этом примере запускается программа-процедура, код которой хранится по внешнему имени // транслятор. В качестве параметров транслятору передаются два файла // ТЕКСТ и // КОД. Используя эти фактические параметры (конкретные файлы), транслятор считывает текст и формирует необходимый код. В этих примерах используются внешние имена. Их создание и привязка к конкретному файлу производятся соответствующими языковыми конструкциями, различными по изобразительным средствам в различных языках.

Основные функции операционной системы. Операционная система расширяет функциональные возможности аппаратуры. Например, для реализации операции a+в транслятору достаточно команд, которые «умеют» выполнять ЦП.

Для реализации, например, генератора массива (выделение области памяти под массив) необходимо обратиться к соответствующей процедуре ОС. Итак, одной из функций ОС является поддержка для реализации языковых средств программирования: создание и уничтожение сложных объектов, таких, как массивы, файлы, справочники, реализация операции над такими объектами, например, чтение и запись в файл (операции обменов с ВУ), запуск программ по внешнему имени, связывание внешнего имени и файла и т. п.

Второй важнейшей функцией ОС является распределение ресурсов вычислительной системы. Дело в том, что реальные ресурсы машины ограничены, а в данный отрезок времени в машине могут находиться в стадии исполнения несколько задач.

Какие же ресурсы распределяет ОС? К важнейшим из них относятся время работы ЦП и оперативная и внешняя память. Кроме этих двух основных функций, ОС предоставляет средства для организации входа пользователя в систему, отрабатывает сбои и предоставляет средства для восстановления вычислительного процесса, собирает статистику о работе всех устройств и интенсивности их использования, обеспечивает защиту данных на основе аппаратных средств, реализует виртуальные возможности для массивов и программ.

Виртуальность — важное качество, определяющее автоматическое перемещение массивов и кодов процедур между оперативной памятью и магнитными дисками, если это необходимо. Таким образом, если происходит обращение к массиву или процедуре, которые в данный момент находятся на магнитном диске, то происходит автоматическое чтение их в оперативную память. Возможно, при этом происходит перемещение каких-то других массивов или процедур из оперативной памяти на магнитный диск.

Как же воспользоваться средствами, реализованными в рамках операционной системы? Очевидно, для этого необходимо уметь вызывать ту или иную процедуру ОС из всего набора. Коды основных процедур всегда находятся в оперативной памяти, куда они перемещаются в процессе инициализации системы. Кроме процедур, в памяти находятся переменные ОС и массивы, где организованы различного рода таблицы, списки необходимой информации. Некоторые массивы существуют всегда, некоторые создаются и уничтожаются в процессе работы системы.

Для вызова процедур ОС в центральном процессоре существуют специальные команды. Например, возможна такая операция «ВХОДВОС», где *n* — порядковый номер процедуры ОС. Конечно, с номерами работать неудобно, поэтому, например, для всех трансляторов можно предусмотреть массив, где устанавливается соответствие между некоторым идентификатором и номером соответствующей процедуры ОС. Такой массив называется «интерфейс» с ОС и является постоянным контекстом трансляции. Имена всех интерфейсных процедур известны, и их можно явно указывать в программе. Например, ... запуск задачи (//КОДПРОГР). В этом примере осуществляется запуск программы с внешним именем //КОДПРОГР как некоторой задачи. Таким образом, из программ можно явно обращаться к процедурам ОС для реализации тех или иных функций. Другой возможный способ обращения к ОС — по прерыванию. Прерывание — это нарушение естественного хода выполнения команд программы и запуск необходимой процедуры. Сигналы прерываний могут возникать по различным причинам. Например, по окончании некоторого обмена информацией, который автономно проводил процессор ввода-вывода, или после нажатия соответствующей клавиши на дисплее, или по сигналу от некоторого внешнего объекта, работой которого управляет вычислительная машина. В последнем случае в оперативной памяти отсутствует код или массив (в момент обращения) для организации чтения информации с маг-

нитного диска. В разных системах прерывания реализуются по-разному. В некоторых, например, управление всегда передается специальной процедуре, а на специальный регистр ЦП заносится номер прерывания для последующего анализа и исполнения необходимой реакции. В других существует специальный массив — коммутатор прерываний, где находятся указатели на процедуры реакции (метки процедур). В этом случае по номеру прерывания ЦП обращается к коммутатору прерываний, выбирает соответствующую метку и запускает необходимую реакцию. После обработки прерывания, как и после запуска и завершения работы любой процедуры, управление возвращается в точку программы, где было вызвано прерывание.

Рассмотрим теперь пример входа с дисплея некоторого пользователя в систему и предоставляемые ему возможности для работы. На этом примере поясним некоторые функции ОС.

При входе пользователя в систему с некоторого терминала ОС создает задачу и запускает в рамках этой задачи программу с фактическим параметром — указатель на файл-терминал. Используя этот параметр, запущенная программа может исполнять операции чтения информации (приказов) с терминала, разбирать их (анализировать текст) и запускать необходимые процедуры. Например, транслятор, редактор текста, исполнение некоторой ранее созданной программы по внешнему имени, создание файлов и занесение их в архив, т. е. установление соответствия между созданными файлами и некоторыми внешними именами и т. п.

Все эти действия осуществляются явным запуском необходимых процедур ОС или запуском системных программ по внешним именам из архива системы, доступного всем пользователям. Например, транслятор, редактор, вывод текста на печать и т. п. Это все возможно, когда в системе уже организована задача и пользователь набирает необходимые приказы.

Следует отметить, что в рамках задачи может быть организована и некоторая параллельная деятельность — созданы процессы. Дело в том, что некоторая программа или процедура может быть запущена для исполнения или последовательно, или параллельно отдельным процессом. Возникает вопрос: «Что значит параллельные, если в системе только один процессор?» Параллельная обработка задач или процессов осуществляется разделением по времени работы центрального процессора, т. е. по некоторому круговому алгоритму каждому процессу выделяется квант времени, после которого происходит переключение на другой процесс. Естественно, что такое же переключение происходит, если процесс ожидает окончания обмена с внешним устройством.

Прежде чем вернуться к анализу организации входа пользователя в систему, следует конкретизировать понятия задачи и процесса.

Относительно системы задача — это некоторый вид параллельной деятельности по исполнению программы от лица конкретного пользователя в рамках выделенных ему ресурсов (бюджет) по времени ЦП, объему требуемой внешней памяти и т. п.

Одна из таких задач — «паспорт задачи», который служит массивом ОС и где хранится информация о текущем состоянии задачи. В рамках задачи могут исполняться несколько процессов. Представителем процесса тоже будет область памяти, где находится информация о состоянии процесса, например информация, необходимая для возобновления процесса исполнения на ЦП.

Итак, вернемся к задачам организации входа пользователя в систему. Все пользователи системы зарегистрированы в специальном файле, расположенном на магнитном диске. В этом файле для каждого пользователя, кроме его имени, указывается пароль для входа, выделенные ему бюджеты ресурсов (время ЦП, объем памяти на магнитном диске и т. п.), которые он не имеет права превышать, ссылка на его так называемый корневой справочник, элементами которого являются внешние имена и ссылки на файлы. Кроме такого файла «всех пользователей», в таблицах операционной системы хранятся указатели на все задачи (таблица задач) и таблица всех устройств, где

отражено текущее состояние.

При первоначальном нажатии специальной клавиши в машину поступает сигнал, который, например, вызывает прерывание ЦП, и в определенное место памяти, связанное с номером дисплея, заносится информация о том, что дисплей стал активным.

Соответствующая процедура отработки прерывания создает в рамках некоторой всегда существующей системной задачи процесс знакомства с пользователем. Здесь как параллельная деятельность запускается специальная программа или процедура ОС, которой в качестве параметра передается указатель на файл-терминал.

Используя этот указатель, запущенная процедура требует от пользователя указать свое имя и пароль (запись на терминал) и ждет (чтения с терминала), когда пользователь ответит. Анализируя полученный ответ, процедура ОС сравнивает его с информацией из «файла пользователей» и в зависимости от результата создает или нет задачу конкретного пользователя, в рамках которой происходит его дальнейшая работа, т. е. исполнение различных программ. Здесь возможны явное или по прерыванию обращение к процедурам ОС для создания массивов, файлов, задач, процессов, организации виртуальных свойств, реализации операций чтения и записи для файлов, приостановки или уничтожения задач и процессов; переключение работы ЦП с одного процесса на другой по исчерпанию кванта времени или по другому событию и т. п.

Все обсуждения, приведенные выше, никак не связывались с особенностями супер- или мини- и микро-ЭВМ. С точки зрения пользователей, создающих различного рода прикладные пакеты, например, для решения научно-технических задач эти особенности не должны быть существенными, так как комфортабельная работа необходима на всех машинах любой производительности. Эти особенности по характеру не должны отличаться от особенностей, связанных с работой на различных ЭВМ (не обязательно супер) одинаковой производительности, но построенных по разным принципам. Относительно ОС и языков программирования (по крайней мере, базового языка) такие особенности необходимо учитывать.

Что же это за особенности? Любые супер-ЭВМ очень критичны, например, к переключению процессора с одной задачи на другую и, вообще, к различного рода прерываниям. В этом случае скорость вычислений резко падает. Это связано, например, с наличием в очень быстрых машинах буферной памяти для размещения данных и команд. Наличие такой буферной памяти позволяет ЦП только при первом обращении к какому-то слову оперативной памяти работать со сравнительно медленной скоростью, а затем использовать его копию из быстрого буфера. Переключение ЦП на другую деятельность и возобновление прерванных вычислений может вновь потребовать обращения к «медленной» оперативной памяти. Обсудим следствия.

В некоторых системах, например, в комплексе с основной сверхбыстрой машиной работает средняя машина-сателлит, на которой ведется редактирование текстов, подготовка кода и необходимых данных. Затем эти данные передаются в основную машину, где осуществляется «производственный» счет. Благодаря этому, например, сокращается количество прерываний от терминала и основной вычислитель разгружается от работы, требующей частого переключения ЦП с одной деятельности на другую. Сократить количество таких переключений можно и по-другому. Например, организовать работу с внешними устройствами не по прерыванию, а осмотром и анализом деятельности аппаратуры в удобные моменты времени. Дело в том, что любая «счетная» задача в каких-то точках программы требует обмена информацией с ЗУ или для чтения данных и для записи результатов. Конечно, существуют асинхронные средства обменов, но обязательно есть момент, когда необходимо ждать окончания обмена. Этот момент и «удобен» для обработки различных внешних сигналов, например, при завершении каких-то обменов — в этот момент все равно необходимо переключение работы ЦП на другой процесс или задачу.

К сожалению, во многих супер-ЭВМ отказываются от реализации виртуальных механизмов, так как это связано с более сложной обработкой адресов переменных и реализацией сложного механизма автоматической «откачки-подкачки» необходимой информации. Такой отказ от реализации виртуальных механизмов накладывает существенные ограничения на программирование и усложняет процесс создания необходимых программ.

Для трансляторов существенным являются архитектурные особенности супер-ЭВМ. В настоящий момент можно выделить два подхода — векторизация и распараллеливание исполнения операций. Векторизация — это введение специальных операций над векторами. Например, пусть есть три массива, и следует каждый элемент первого массива сложить с соответствующим элементом второго, а результат записать в тот же элемент третьего массива.

Это и есть пример векторной операции. Такие операции могут выполняться очень быстро с использованием так называемого «конвейера». В этом случае в стадии исполнения находятся сразу несколько элементов векторов. Например, для какого-то элемента записывается результат, а какие-то только считываются из памяти. Используя такой принцип, можно получить скорость исполнения одной операции над соответствующими элементами векторов, равную тактовой частоте машины. За один такт в ЦП осуществляется много элементарных действий типа «передача информации регистр — регистр». В этом случае для машины с тактовой частотой 10^{-8} с можно достичь скорости для векторных операций $100 \cdot 10^6$ операций в секунду.

Другой подход — возможность параллельного исполнения операций. В этом случае в ЦП имеется несколько исполнительных устройств, каждое из которых умеет выполнять определенный набор операций. Например, два устройства для сложения, два — для умножения, одно выполняет логические операции и т. п. Каждое из устройств тоже может обладать свойствами «конвейера», но не по исполнению операции над вектором, а по исполнению операций над не связанными векторной структурой данными (скалярные вычисления). В этом случае, если для какой-то операции «*» необходим результат ранее исполняемой команды «+», то операция должна поступить на исполнение в ЦП только после завершения соответствующей операции«+».

Оба этих подхода для получения максимальной производительности ставят сложные требования к трансляторам. Если не рассматривать сравнительно простой случай явного использования векторной операции, то эти требования для векторных машин сводятся к автоматическому выявлению на стадии трансляции неявных векторных структур для использования быстрых векторных операций. И во втором случае на стадии трансляции необходимо выявить независимые по исполнению операции для организации их параллельной работы на различных исполнительных устройствах.

Перечисленные проблемы не относятся исключительно к супер-ЭВМ, однако их проявление стало актуальным именно в процессе поиска новых путей для повышения производительности.

БК ЗА РОГА

С. ГВОЗДЕВ, Г. ЭРНЕСТСОНС

Использование только ФОКАЛа иногда препятствует эффективному решению задач на ЭВМ. Это особенно проявляется при программировании динамических графических изображений или при управлении внешними устройствами. Некоторые задачи вообще невозможно решить, оставаясь в рамках ФОКАЛа, например, создание алфавита пользователя или программирование звуковых эффектов. Однако с помощью ФОКАЛа можно создавать разнообразные программы как чисто вычислительного, так и игрового характера, используя при этом графические возможности и доступ к порту ЭВМ.

Использование плавающей арифметики ФОКАЛа в БК-0010

Микро-ЭВМ «Электроника БК-0010», получившая у нас большое распространение, оснащалась до последнего времени интерпретатором с языка ФОКАЛ[1], выполненным в микросхеме К1801РЕ2-018 или К1801РЕ2-084 в комплекте с классом КУВТ-86.

Применение подпрограмм пользователя в рамках так называемого FOCODa (автор В. Н. Андреев, Москва) для названных задач не всегда дает желаемый результат. Для этого предпочтительнее программировать исключительно в кодах ЭВМ, полностью отказываясь от ФОКАЛа, получая в качестве компенсации в свое распоряжение все возможности БК-0010.

Программирование в кодах позволяет не только повысить скорость исполнения программы, но и сэкономить память, тем более что этот ресурс в БК-0010 крайне ограничен.

Однако, отказываясь от ФОКАЛа, приходится считаться с отсутствием в системном программном обеспечении ЭВМ арифметики чисел с плавающей запятой и библиотеки подпрограмм для вычисления стандартных функций.

Вниманию читателей предлагается простой способ доступа на уровне кодов ЭВМ к находящимся в ПЗУ ФОКАЛ-арифметике и встроенным функциям ФОКАЛа. В этом случае удается сэкономить память ЭВМ, затраты которой были бы значительными, если в нее вводить собственный вариант программ для работы с числами с плавающей запятой. Так, объем программ в ПЗУ, реализующих ФОКАЛ-арифметику, составляет 2,7 кбайт.

Для иллюстрации рассмотрим пример, часто встречающийся при сборе данных. Предположим, что некоторое устройство, например измеритель температуры, подключено к порту БК-0010 и по приказу ЭВМ выдает на него двоичные числа, пропорциональные значению температуры. Таким приказом является установка младшего бита порта на короткое время в единицу. Пусть необходимо произвести 1000 измерений и поместить их значения в массив T(*i*). Программа на ФОКАЛе, производящая указанные действия, будет выглядеть следующим образом:

```
10.10 X FP (1,177777); C Очистка регистра вывода
10.20 S N=1000; C Установка числа измерений
10.30 F I=1,N;X FP(2,1); H FP(1,1); S T(  )=FP(0,
177777)
10.40Q
```

В этой программе около 95% машинного времени уходит на интерпретацию программы. Соответствующая программа в машинных кодах работает в 40—50 раз быстрее. Кроме увеличенной скорости, мы имеем выигрыш в памяти. Дело в том, что для хранения числа в ФОКАЛе отводится 8 байтов, в то время как фактически число размещается в 4 байтах. Таким образом, выигрыш при хранении данных равняется 4 байтам на каждое число.

Отметим, что выигрыш в скорости тем значительнее, достигая 2—3 порядков, чем более просты математические операции, которые используются в ФОКАЛ-программе. Однако даже для вычисления сложных функций, типа FSIN (X), программа в кодах работает в 1,5 раза быстрее.

Программу в кодах, аналогичную ФОКАЛ-программе, мы приведем ниже, после разбора арифметики чисел с плавающей запятой.

ФОКАЛ-арифметика

В связи с отсутствием в публикациях сведений о ФОКАЛ-арифметике дадим краткий обзор на эту тему. Будем считать, что читателю известна архитектура ЭВМ и элементы программирования на АССЕМБЛЕРЕ (см., например, [2]).

В ФОКАЛЕ реализована арифметика вещественных чисел с плавающей запятой одинарной точности. В памяти ЭВМ числа представлены в двухсловном формате. Порядок записывается в одном байте, а мантисса занимает 3 байта. Отрицательный порядок и мантисса кодируются в дополнительном коде.

младшие разряды мантиссы	порядок
старшие разряды мантиссы	

Знак мантиссы

Арифметические операции с числами производятся в трехсловном сумматоре, реализован-

ном программно и размещенном в памяти ЭВМ в адресах 1710, 1712, 1714.

Кодирование операций осуществляется с помощью псевдокоманд, при встрече которых в тексте программы происходит прерывание по вектору 10 (прерывание по коду резервной команды) и управление передается на обработку этих команд.

Арифметические операции в ФОКАЛе можно условно разделить на двухместные (сложение, умножение) и одноместные (целая часть, абсолютная величина).

Первым операндом в двухместных операциях является содержимое сумматора, второй операнд выбирается из памяти ЭВМ в зависимости от метода адресации. Результат операции всегда помещается в сумматор.

Двухместная операция кодируется четырехзначным восьмеричным числом 70XY, где X и Y — восьмеричные цифры, означающие: X — код операции; Y — метод адресации операнда (табл. 1,2). При описании арифметических операций и функций используются следующие обозначения: S — сумматор, Op — операнд.

Таблица 1

Код X	Операция	Мнемоника
0	op — > S	FGET
1	S + op — > S	FADD
2	S – op — > S	FSUB
3	S / op — > S	FDIV
4	S * op — > S	FMUL
5	S Λ op — > S	FPOW
6	S — > op	FPUT

Типы адресации достаточно очевидны и ясны из обозначений табл. 2. Однако есть нюансы отно-

Таблица 2

Код У	Адрес операнда	Мнемоника
0	Адрес в следующем слове	@#op
1	Адрес в R2	(R2)
2	(R2) +4, адрес в R2	+(R2)
3	Операнд в вершине стека	(SP)
4	Адрес в вершине стека	@ (SP)
5	Операнд в следующих двух словах	# op
6	Не работает	

сительно традиционных методов адресации в архитектуре ЭВМ. Так, использование адресации с кодом 2 (автоинкрементная адресация) имеет ту особенность, что изменение содержимого регистра R2 производится до операции, а не после, как обычно при выполнении машинных команд. При размещении операнда в стеке (адресация с кодом 3) программист сам должен следить за указателем стека.

Псевдокоманды с кодом 707Y задают одноместные операции, выполняющиеся над содержимым сумматора (табл. 3).

При выполнении псевдокоманд содержимое регистров сохраняется, кроме тех случаев, когда изменение содержимого регистров обусловлено самой псевдокомандой. Необходимо отметить, что признаки NZVC после выполнения псевдокоманд не устанавливаются в соответствии с результатом операции.

Проиллюстрируем на примерах использование описанных операций. Программы в кодах ЭВМ приводятся в соответствии с мнемоникой, принятой в АССЕМБЛЕРе. В приводимых примерах значащими являются только столбцы с адресом памяти и содержимым слова по данному адресу. Поля с мнемоникой команд и комментарием приводятся только в целях облегчения понимания.

Таблица 3

Команда	Операция	Мнемоника
7070	Нормализация числа в S	FNOR
7071	Целая часть S — > R1	FINT
7072	Sgn (S) — > S	FSGN
7073	Abs (S) — > S	FABS
7074	— S – > S	FNEG
7075	Перевод числа, заданного символьной строкой в S	FREAD
7076	Вывод S на экран R2 — формат вывода	FPRINT
7077	0 — > S	FZERO

Загрузка сумматора некоторым числовым значением производится с помощью операции FGET:

```
Адрес    Содержимое  Мнемоника      Комментарий
002000   007000      FGET @# A      ; Операция FGET.
002002   002100                     ; Адрес операнда.
-------------------------------------------------------
002100   000002 A:                  ; Число 2.0 в формате
002102   040000                     ; с плавающей запятой.
```

Та же самая операция возможна с адресацией через регистр R2.

```
002000   012702      MOV  # A, R2   ; Загружаем адрес
002002   002100                     ; переменной A  в R2.
002004   007001      FGET (R2)      ; A в сумматор.
```

Адресация через регистр R2 особенно эффективна, когда переменные занимают смежные участки памяти ЭВМ. В этом случае удобно использовать автоматическую модификацию индексного регистра, так называемую автоиндексацию. Рассмотрим пример, в котором применяется автоиндексация. Пусть у нас есть фрагмент ФОКАЛ-программы:

```
10.10   S   C=2.0+3.0
10.20   T   C; Q
```

Программа в кодах может выглядеть следующим образом:

```
002000  012702      MOV #(A-4), R2      ; Адрес массива перемен-
002002  002014                          ; ных с учетом автомати-
                                        ; ческого наращивания.
002004  007002      FGET +(R2)          ; Занести в сумматор 2.0
002006  007012      FADD + (R2)         ; Добавить к сумматору 3.0
002010  007062      FPUT  + (R2)        ; Результат запомнить в C.
002012  005002      CLR R2              ; Подготовиться к выводу.
002014  007076      FPRINT              ; Вывести сумматор на экран.
002016  000000      HALT
002020  000002 A:                       :  2.0
002022  040000
002024  000002 B:                       ;  3.0
002026  060000
002030          C:                      ; Здесь будет результат.
```

В приведенном примере в переменных A и B записаны константы в формате чисел с плавающей запятой. В данном случае перевод в нужный вид был сделан вручную. Иногда такое преобразование можно поручить ЭВМ. Очевидно, что в процессе решения задач возникают ситуации, требующие преобразования чисел из одного вида в другой. Например, целые числа при программировании в кодах гораздо удобнее. Но, перед тем как обратиться к ФОКАЛ-арифметике, целое число надо преобразовать в нормальный двухсловный формат. Целое число, расположенное по адресу A, можно поместить в сумматор при помощи следующего фрагмента программы:

```
002000  012737      MOV    # 17, @# 1710  ; Порядок целого числа
002002  000017                            ; равен 15.
002004  001710
002006  013737      MOV    @# A, @# 1712  ; Записали целое число
002010  002100                            ; в виде ненормализован-
002012  001712                            ; ной мантиссы.
002014  005037      CLR    @# 1714        ; Незначащие разряды в 0.
002016  007070      FNOR                  ; Нормализовали сумматор.
```

В дальнейшем значение сумматора можно или использовать непосредственно в вычислениях, или записать в память ЭВМ.

В практике часто приходится вводить числовые данные с клавиатуры ЭВМ. При этом в памяти ЭВМ образуется последовательность байтов, в которой находится число в символьном виде в коде КОИ-8. Для перевода числа в двоичный вид можно воспользоваться операцией FREAD. Символьная строка предварительно должна быть перекодирована, так как в ФОКАЛе для некоторых символов используются коды, отличные от стандартных кодов КОИ-8. При записи числа, в частности, необходимы символы «—» (минус) и «+» (плюс). В ФОКАЛе для этих символов употребляются восьмеричные коды 202 и 201 соответственно вместо 55 и 53. Признаком конца символьной строки является код 217. В следующем примере с клавиатуры вводится число, коды клавиш «+», «—» и «ВВОД» перекодируются. Затем число переводится в плавающий формат и записывается в сумматор.

```
002000  012701        MOV  # A,R1          ; Поместить адрес символьной
002002  002100                             ; строки в R1.
002004  104016  4¤:   EMT  16              ; Ввести код с клавиатуры в R0
002006  120027        CMPB R0, # ' +       ; Ввели "+"?
002010  000053
002012  001003        BNE  1¤              ; Нет.
002014  112700        MOVB # 201, R0       ; Да. Заменить код.
002016  000201
002020  000413        BR  2¤
002022  120027  1¤:   CMPB R0, # ' -       ; Ввели "-"?
002024  000055
002026  001003        BNE  3¤              ; Нет.
002030  112700        MOVB  # 202, R0      ; Да.
002032  000202
002034  000405        BR  2¤
002036  120027  3¤:   CMPB  R0, #12        ; Нажали "ВВОД"?
002040  000012
002042  001002        BNE 2¤               ; Нет.
002044  112700        MOVB #217, R0        ; Код завершения строки.
002046  000217
002050  110021  2¤:    MOVB  R0, (R1)+     ; Записали символ в буфер.
002052  120027        CMPB  R0,  # 217     ; Если закончили ввод,
002054  000217                             ; то выполнить перевод.
002056  001352        BNE  4¤              ; Иначе ввести следующий символ.
002060  012703        MOV   #A, R3         ; Для перевода надо поместить
002062  002100                             ; адрес символьной строки в R3.
002064  112304        MOVB (R3)+, R4       ; Обязательно поместить первый
                                           ; символ в R4.
002066  112737        MOVB   # 20, @#1723  ; Обязательно заполнить.
002070  000020
002072  001723
002074  007075        FREAD                ; Поместить число в сумматор.
002076  000000        HALT
002100          A:                         ; Буфер ввода.
```

Если символьная строка не требует перекодировки, то для перевода числа в плавающий формат необходимы строки программы, записанные в адресах 2060 по 2074.

Операция FREAD вводит в сумматор всегда положительное число. Если символьная строка задает отрицательное число, то после можно воспользоваться операцией FNEG для правильного представления числа в сумматоре.

В списке операций ФОКАЛа отсутствует операция для перевода числа из плавающей формы в символьный вид. Однако операция FPRINT позволяет вывести на экран содержимое сумматора. Регистр R2 содержит при этом требуемый формат. Восьмеричная константа 4012 соответствует формату ФОКАЛа %8.04. Если R2 равен нулю, то число будет выведено в экспоненциальной форме. При использовании FPRINT необходимо иметь в виду, что операция не сохраняет содержимое сумматора.

Завершим рассмотрение операций ФОКАЛа фрагментом программы в кодах ЭВМ, соответствующей ФОКАЛ-программе, которая приведена в начале статьи.

```
002000  012705          MOV   #1000., R5      ; Число измерений 1000
002002  001750
002004  012702          MOV   #<T-4>, R2      ; Адрес массива измерений.
002006  002052
002010  012701          MOV   #177714, R1     ; Адрес порта ввода/вывода.
002012  177714
002014  005011          CLR   (R1)            ; Очистить регистр вывода.
002016  052711    1¤:   BIS   #1, (R1)        ; Произвести измерение.
002020  000001
002022  042711          BIC   #1, (R1)
002024  000001
002026  012737          MOV  #17, @#1710
002030  000017
002032  001710
002034  011137          MOV  (R1), @# 1712    ; Записать измерение
002036  001712                                ; в сумматор.
002040  005037          CLR @#1714
002042  001714
002044  007070          FNOR                  ; Перевести в плавающий формат.
002046  007062          FPUT  +(R2)           ; Записать в массив T( ).
002050  077516          SOB  R5, 1¤           ; Следующее измерение.
002052  000000          HALT
002054            T:                          ; Массив измерений T.
```

Данная программа иллюстрирует еще один источник экономии памяти при программировании в кодах. Несмотря на довольно длинную запись, программа в кодах занимает в памяти ЭВМ примерно в 2 раза меньше места, нежели чем ее прототип на ФОКАЛе. В среднем, как показывает опыт программирования в кодах, данная пропорция сохраняется для достаточно длинных и содержательных программ.

Встроенные математические функции ФОКАЛа

Обращение к встроенным функциям ФОКАЛа производится как к обычным подпрограммам, в качестве регистра возврата используется счетчик команд PC (или R7), т. е.

JSR PC, <адрес функции в ПЗУ ФОКАЛа>

Адреса встроенных функций для микросхемы К180ИРЕ2=018 приведены в табл. 4. Для микросхемы К180ИРЕ2=084 адреса, естественно, другие. Полный перечень адресов для микросхемы =084 будет приведен в приложении.

Таблица 4

Адрес	Функция	Мнемоника
126550	int(S) — > S	FITR
132260	cos(S) — > S	FCOS
132250	sin(S) — > S	FSIN
132122	кв. корень (S) — > S	FSQT
132550	случ. число — > S	FRAN
126554	sign(S) — > S	FSGN
126540	|S| — > S	FABS
134132	lg(S) — > S	FLOGI0
134156	ln(S) — > S	FLOG
134416	exp(S) — > S	FEXP
133702	tg(S) — > S	FTAN
134632	arctg(S) — > S	FATAN
133656	arccos(S) — > S	FACOS
133402	arcsin(S) — > S	FASIN

При использовании встроенных функций программист должен сам побеспокоиться о сохранении регистров ЭВМ.

Программа начальной инициализации

Опишем сейчас действия, которые надо проделать, прежде чем воспользоваться ФОКАЛ-арифметикой или встроенными функциями. Связь с программным обеспечением, находящимся в ПЗУ, осуществляется через векторы прерывания. Выше уже упоминалось, что при встрече кодов псевдокоманд происходит прерывание по резервной команде (вектор прерывания 10). При обработке псевдокоманд ФОКАЛ использует внутренние обслуживающие подпрограммы. Служебные подпрограммы выполняют такие действия, как работа со строками, ввод-вывод символьной информации, вычислительные действия, выдача сообщений об ошибках. Часть служебных подпрограмм вызывается с помощью TRAP-запросов. Вектор

прерывания для TRAP-запросов содержится в ячейке 34.

Следовательно, для успешной работы с программным обеспечением, размещенном в ПЗУ ФОКАЛа, необходимо создать соответствующую рабочую среду. В принципе если программа пользователя написана без единой ошибки, то для правильной обработки псевдокоманд достаточно перед их выполнением заменить содержимое ячеек 10 и 34 на адреса связи с ФОКАЛом 125636 и 121136 соответственно. Однако в этом случае программа пользователя беззащитна перед любыми ненормальными ситуациями, происходящими при работе обслуживающих подпрограмм ФОКАЛа. Например, невозможно гарантировать, что в процессе решения задачи не возникнет попытка деления на нуль, или попытка извлечь корень из отрицательного числа, или попытка перевести из символьного в двоичный формат нечисловую информацию. В этом случае при любой ошибке управление передается ФОКАЛу, который в «наказание» стирает как программу пользователя, так и все данные.

Предлагаемая вспомогательная программа осуществляет корректный интерфейс между программой пользователя в кодах ЭВМ, обслуживающими подпрограммами ФОКАЛа и монитором, из которого был произведен запуск программы пользователя. В любом случае вход в программу пользователя происходит из вспомогательной программы, которую мы будем называть в дальнейшем программой начальной инициализации ПИФ.

Основные функции программы ПИФ:

— сохранение области векторов прерывания монитора, из которого производится пуск программы;

— создание необходимой рабочей среды для ФОКАЛа;

— перехват всех TRAP-запросов и анализ их корректности;

— обработка ошибочных TRAP-запросов;

— возврат в монитор, из которого был произведен запуск программы.

Область памяти 1000 — 1777 используется программой ПИФ и рабочими ячейками ФОКАЛа. Приведем распределение адресного пространства.

1000	Стартовый адрес программы. Сохранение области векторов прерывания, заполнение рабочих ячеек ФОКАЛа, установка векторов прерывания, установка указателя стека.
1060	«Ловушка» для TRAP-запросов.
1120	Обработка недействительных TRAP-запросов.
1152	Выход в вызывающий монитор: восстановление векторов прерывания, сообщение о месте останова.
1200	Адреса обработки TRAP-запросов с 200 по 262.
1264	Резерв для использования TRAP с 264 по 376.
1400	П/П для сохранения и восстановления векторов прерывания; область сохранения.
1500	Рабочие ячейки ФОКАЛа.
2000	Первая команда программы пользователя.

Программа пользователя в кодах должна размещаться в памяти ЭВМ, начиная с адреса 2000. Более того, по адресу 2000 должна располагаться первая исполняемая команда программы. Однако пуск программы производится через ПИФ, т. е. с адреса 1000. Завершаться программа пользователя может традиционной командой HALT. При этом адрес останова находится в регистре R5. Естественно, результат при нажатии на клавишу «СТОП» будет точно такой же.

Выше речь уже шла об ошибочных ситуациях при исполнении операций ФОКАЛа или при вычислении стандартных функций. В этом случае ПИФ обеспечивает нормальный останов программы с выдачей дополнительно некоторого диагностического сообщения. Например, если произошла ошибка деления на нуль, то на экран выдается:

```
ПОПЫТКА ДЕЛЕНИЯ НА НУЛЬ
002450 : 104635
```

Для недопустимых TRAP-запросов выдается только вторая строка. В обоих случаях программа завершает работу таким же образом, как описано выше.

Четные номера TRAP-запросов с кодами 264—376 свободны и могут быть использованы программистом для обращения к своим подпрограммам с помощью программного прерывания TRAP. Область памяти с адресами 1264—1376 зарезервирована для соответствующих адресов подпрограмм пользователя. Если подпрограмму, расположенную начиная с адреса SUBPR, пользователь захочет вызвать с помощью, например, запроса TRAP 264, то предварительно необходимо адрес подпрограммы поместить в слово с адресом 1264. Возврат из подпрограммы производится через регистр R5.

```
        MOV #SUBPR, @#1264   ; Адрес.подпрограммы
                             ; пользователя в таблицу.
                             ; TRAP – запросов.
-------------------------------------------------------
        TRAP 264             ; Вызов подпрограммы.
-------------------------------------------------------
SUBPR:                       ; Начало подпрограммы.
        RTS R5               ; Возврат.
```

Кроме того, программист может включать в свою программу и некоторые служебные TRAP-запросы ФОКАЛа, в частности запросы с номерами 0—177, которые выдают на экран символ, код которого соответствует номеру запроса.

Для тех, кто захочет воспользоваться средствами, описанными в данной статье, мы приводим текст ПИФ. Для экономии места дается только образ памяти в восьмеричном виде. Пользователь должен будет ввести эту программу в память ЭВМ с помощью любого монитора, начиная с адреса 1000.

Текст программы начальной инициализации ПИФ

```
004767 000374 012737 125636 000010 012737 001060 000034
012767 001032 000210 000137 120056 012706 001000 012737
001152 000004 012767 001120 000164 104004 000137 002000
106466 000002 010566 000002 011605 016516 177776 106216
100402 000137 121502 103404 106116 162716 103400 013607
012706 001000 004437 110346 014501 103013 104412 016101
014211 005002 104020 000407 011605 012706 001000 004437
110346 004767 000232 004437 110362 000000 000000 000000
121572 121624 122134 122074 132064 132052 122360 001120
121520 121640 121752 001120 124134 001120 122250 001120
001120 122012 122040 125326 122056 122022 121556 125556
001120 001120 000000 000000 000000 000000 000000 000000
000000 000000 000000 000000 000000 000000 000000 000000
000000 000000 000000 000000 000000 000000 000000 000000
000000 000000 000000 000000 000000 000000 000000 000000
000000 000000 000000 000000 000000 000000 000000 000000
005000 012701 001430 012705 000040 112021 077202 000207
005001 012700 001430 000767
```

В заключение отметим,что указанный подход применим и в том случае, когда используется микросхема с другим языком, например, с БЕЙСИ-Ком. Правда, в этом случае можно воспользоваться только идеей. К сожалению, ни завод-изготовитель, ни разработчики языков не представляют пользователям ЭВМ достаточной информации о программе, записанной в ПЗУ. Это приводит к тому, что пользователи вынуждены проводить свои исследования, подобно этому.

ПРИЛОЖЕНИЕ

Приложение содержит все изменения, которые надо внести в текст программы ПИФ или учесть при использовании ФОКАЛа, находящегося в микросхеме К1801РЕ2-084

1. Адреса встроенных функций для микросхемы К1801РЕ2-084.

Таблица 5

Адрес	Функция	Мнемоника
126672	int(S) — > S	FITR
132262	cos(S) — > S	FCOS
132252	sin(S) — > S	FSIN
132124	кв. корень (S) — > S	FSQT
132552	случ. число — > S	FRAN
126666	sign(S) — > S	FSGN
126662	ISI — > S	FABS
134134	lg(S) — > S	FLOGI0
134160	ln(S) — > S	FLOG
134420	exp(S) — > S	FEXP
133704	tg(S) — > S	FTAN
134634	arctg(S) — > S	FATAN
133660	arccos(S) — > S	FAGOS
133404	arcsin(S) — > S	FASIN

2. Значение векторов прерывания 10 и 34 должны быть равны 125760 и 121300 соответственно.

3. В программе ПИФ должны быть изменены значения ячеек 1006, 1104 и 1140 на 125760, 121654 и 014313.

4. В программе ПИФ изменяется таблица TRAP-запросов, расположенная начиная с адреса 1200. Для микросхемы -084 значения этой таблицы следующие:

```
121744 121776 122306 122246 132012 132000 122532
001120 121672 122012 122124 001120 124256 121414
122622 001120 001120 122164 122212 125450 122230
122174 121730 125700
```

Список литературы

1. Писаревский А. Н. Осетинский Л. Г., Осетинский М. Г. ФОКАЛ — диалоговый язык для мини-ЭВМ. — Л.: Машиностроение, 1985. — 195 с., ил.

2. Вигдорчик Г. В., Воробьев А. Ю., Праченко В. Д. Основы программирования на ассемблере для СМ ЭВМ. — М.: Финансы и статистика, 1983. — 256 с., ил.

ПРОГНОЗ РАЗВИТИЯ ЭВМ

До недавнего времени однопроцессорные ЭВМ составляли большую часть парка ЭВМ, однако в последние годы ситуация изменяется в результате разработки как мини-супер-ЭВМ, так и многопроцессорных супер-мини-ЭВМ. Первые представляют собой компактные и упрощенные супер-ЭВМ со средствами векторной обработки и ускоренной арифметической обработкой чисел в форме с плавающей запятой. Вторые — системы слабо связанных процессоров, в которых процессоры делят общую магистраль и память. В целом ожидается, что к 1990 г. по мере подключения к базовым магистралям все большего числа средств, характерных для супер-ЭВМ, граница между мини-супер-ЭВМ и супер-мини-ЭВМ будет размываться.

Супер-мини-ЭВМ будут включать все большее число процессоров. Если число процессоров в супер-мини-ЭВМ за ближайшие два года едва ли превысит 4—8, то число процессоров в будущих мини-супер-системах достигнет 20—30. Независимо от применяемой «элементной базы» все супер-мини-ЭВМ будут также включать емкую (до 1 Гбайт) память произвольного доступа, буфер данных емкостью до 64 Кбайт и каналы процессор—память с пропускной способностью свыше 100 Мбайт/с. Емкость дисковой памяти превысит многие гигабайты.

Получит распространение новый вид супер-мини-ЭВМ — рабочие станции с многопроцессорной архитектурой (так называемые суперстанции), ориентированные на обработку изображений, трехмерную графику, задачи искусственного интеллекта и соответствующие комбинированные задачи.

Другое новое направление исследований и разработок в этой области — слияние архитектурных свойств мини-супер-ЭВМ и супер-мини-ЭВМ, в частности, появление в таких системах очень емкой памяти произвольного доступа, включение конвейерных средств обработки команд и данных, применение процессоров с упрощенной системой команд (так называемые PISC-процессоры), параллельных АЛУ, магистралей данных 64-разрядного формата, средств балансирования нагрузки за счет группировки процессоров и устройств, а также применение разнообразных сопроцессоров.

Один из приемов повышения производительности обычных универсальных ЭВМ — применение оборудования с «массивным параллелизмом», в котором ряд команд (загрузка, запоминание, передача и др.) исполняются одновременно. Однако ряд других задач требуют другого подхода, основанного на использовании архитектуры с широкоформатными словами, положенной в основу машины фирмы Multiflow Computer Corp. (Branford, США). Это — 64-разрядная универсальная ЭВМ со специальным длинноформатным компилятором, анализирующим операторы ФОРТРАНа с целью выделения операций, которые можно «собрать» в одну длинную команду и выполнить за один такт. Это могут быть простейшие команды загрузки и запоминания, а также умножения и деления чисел в форме с плавающей запятой. Поскольку вся работа практически осуществляется автоматическим компилятором, машина Multiflow требует сравнительно простого программирования.

Другая машина с оптимизирующим компилятором, использующим внутренний параллелизм оборудования, разработана фирмой Cydrome Inc. (Milpitas, США). В том же направлении работает фирма Hewlett-Packard (HP), используя однако PISC-архитектуру в системах HP-3000 (модели 930 и 950); на этом пути уже достигнуто 2—2,5-кратное увеличение производительности по сравнению с исходной системой HP-3000; так, модель 950 дает производительность 7 млн. операций в 1 с.

Программная совместимость при переходе от младших к старшим моделям ЭВМ — важнейший фактор успешного внедрения новых ЭВМ. Так, фирма DEC в этом отношении имеет ряд преимуществ перед IBM, поскольку все машины VAX работают с одной и той же операционной системой VMS и с теми же прикладными программами. В отличие от этого фирма IBM предлагает различные операционные системы для младших, средних и старших семейств ее машин и не дает ясной возможности постепенного наращивания мощностей соответствующих вычислительных систем. Однако программная совместимость мешает фирме DEC переходить к новым, более эффективным архитектурным решениям.

Машина Butterfly фирмы BBN Advanced Computers Inc. (США) представляет собой непрерывно перестраиваемую многопроцессорную систему, аналогичную известной Connection Machine фирмы Thinking Machines Inc. (США). Машина Butterfly может включать до 256 процессоров 68020, объединяемых через матричный переключатель. Большие надежды возлагаются на рабочие станции с производительностью 10—20 млн. оп/с или 3—10 млн. операций над числами в форме с плавающей запятой в 1 с. Графический кристалл в такой машине может генерировать до 4096 цветов на экране форматом 2 К×2 К элементов изображения. Такая машина разрабатывается фирмой Stellar Computer Inc. (США). Ряд других фирм также ведут интенсивные исследования и разработки в области рабочих станций с расширенными графическими возможностями и повышенным быстродействием. Так, фирма HP совершенствует свою станцию 4129; в новой системе, названной Renaissance, использована схемная реализация многих алгоритмов преобразования трехмерных координат в двумерный образ с помощью НМОП-СБИС, что дает 20-кратное повышение производительности по сравнению с машинами аналогичной стоимости.

По материалам ВИНИТИ

справки, сообщения, реклама

ПРОГРАММНЫЕ СРЕДСТВА — НАСЕЛЕНИЮ (ДЛЯ БК 0010)

Предлагаем читателям краткие сообщения по программам бытового назначения, которые можно приобрести через центры информатики ГКВТИ СССР. Перечисленные программные средства входят также в специализированный фонд алгоритмов и программ НПО «Казань» ВТИ и МЭВЦ «Элекс». Предприятия и организации для приобретения программных средств должны направить гарантийное письмо в адрес предприятий-изготовителей:

1) 420044, Казань, пр. Х. Ямашева, 36. НПО «Казань» ВТИ.

2) 103051, Москва, Малый Сухаревский пер., 9а, МЭВЦ.

В последующих выпусках будут приведены сведения о программных средствах для ПЭВМ «Микроша», «Агат», «Электроника-85», «Роботрон-1715», «Корвет».

РЕКЛАМА (ФОКАЛ). Программа знакомит с техническими характеристиками и графическими возможностями компьютера.

ФАРАОН (ФОКАЛ). На время игры вы становитесь кладоискателем. Вам предстоит отыскивать сокровища в лабиринтах старинных пирамид.

ЛУННИК (ФОКАЛ). Вы — командир космического корабля. Вам предстоит посадить корабль на поверхность Луны.

РОБИНЗОН (ФОКАЛ). Эта игра для любителей острых ощущений. Вам дается возможность, очутившись на необитаемом острове, проверить себя в экстремальных условиях, попробовать победить чудовищ, заселяющих этот остров.

РЕВЕРС (ФОКАЛ). Это древняя логическая игра, заключающаяся в окружении фишек противника своими. Игра предназначена для двух человек, компьютер выступает в качестве анализатора ситуации.

МУЛЬТИК (ФОКАЛ). С помощью этой программы вы можете создать мультипликационный фильм и записать его на магнитную ленту.

ЛАБИРИНТ (ФОКАЛ). Вам предстоит, надеясь только на свою интуицию и везение, выбраться из неизвестного лабиринта, полного тайн и опасностей.

ИППОДРОМ (ФОКАЛ). Эта динамическая игра предлагает вам окунуться в атмосферу гонок и испытать свое везение и интуицию.

БИОРИТМЫ (ФОКАЛ). С помощью этой программы вы сможете определить точный срок, когда, к примеру, можно приступить к выполнению сложной работы — ЭВМ выдаст вам графики ваших биоритмов.

КАЛАХ (ФОКАЛ). Это древняя восточная игра в камешки. Перекладывая камешки по очереди с компьютером, вы приобщаетесь к восточной мудрости.

КУБИК (ФОКАЛ). ЭВМ отправляет вас путешествовать по неведомым граням суперкубика. Будьте готовы к неожиданностям и трудностям.

ТАНКИ (ФОКАЛ). Вы — командир бронетранспортера, который вступает в борьбу с вражескими танками. От вашего умения вести бой зависит исход сражения.

ТУРИСТ (ФОКАЛ). С помощью этой увлекательной и динамичной игры вы, сидя за экраном монитора, сможете пройти по труднодоступным туристским тропам.

ЛОТО-86 (ФОКАЛ). Программа поможет вам выработать в себе такие качества, как внимание, выдержку, быструю реакцию на мгновенно изменяющуюся ситуацию.

ТЕТРИС (ФОКАЛ). Вам нужно успеть аккуратно сложить в стакан быстро падающие фигуры.

ПСИХОТЕСТ (ФОКАЛ). Программа поможет вам разрешить некоторые стрессовые ситуации. Как вы себя чувствуете? Вам нужно отдохнуть? Заняться любимым делом? Разумный совет вам даст программа «Психотест».

МАРС 2 (машинные коды). Это динамическая игра, в которой компьютер назначает вас командиром звездолета. Вашу галактику атакуют звездные завоеватели. Вы получаете приказ спасти людей.

ШАХМАТЫ (машинные коды). Предлагаем вам сразиться в шахматы с компьютером. Он может играть как в силу перворазрядника, так и в силу начинающего шахматиста.

ФОРТ (машинные коды). В этой увлекательной игре компьютер предлагает вам роль командира вертолета. Вам, управляя машиной, нужно обеспечить безопасную высадку десанта.

ПАКМАН (машинные коды). В этой очень динамичной игре от вас потребуются трезвый расчет и хорошая реакция. Скрываясь от преследующих вас привидений, постарайтесь «съесть» как можно больше крестиков.

МЕЛОМАН (машинные коды). Музыкально-игровая программа, с помощью которой вы станете композитором и первым слушателем своих произведений.

КЛАД (машинные коды). Эта азартная многосерийная игра для детей и взрослых. Ваша задача — найдя ключ, убежать от сторожей, которые вас активно ловят.

ЗАМОК 1 (машинные коды). В ходе этой увлекательной игры вам нужно помочь электронному человечку найти клад в за́мке, полном опасностей.

ВИДЕО (машинные коды). Это пакет игровых программ, с помощью которых вы сможете сыграть в футбол, теннис и сквош.

ДОМ (машинные коды). С помощью этой программы дети и взрослые смогут нарисовать забавные картинки.

ХАММУРАПИ (ФОКАЛ). Только компьютер может предоставить вам возможность управлять государством! Испытайте свои силы в этом деле, и вы увидите, как это непросто!

ПИФ-ПАФ (машинные коды). Игра развивает меткость и реакцию. Это увлекательное сражение со сказочными чудовищами.

МЕНЕДЖЕР (ФОКАЛ). Это так называемая деловая игра, в процессе которой вы сможете распоряжаться немалым капиталом.

ЛУЧ (машинные коды). Это игра для двоих. Управляя лучом на экране, постарайтесь пройти больший путь, чем ваш соперник.

МИНЫ (ФОКАЛ). Эта игра для нескольких участников. Полагаясь на свою интуицию, пройдите вслепую по минному полю. Кому из вас больше повезет?

ЗОМБИ (ФОКАЛ). Игра дает возможность проверить себя в экстремальных условиях. Оказавшись в одной клетке с хищниками, проявите хитрость и отвагу.

БЕЙСИК (ФОКАЛ). Интерпретатор языка БЕЙСИК, с помощью которого на БК 0010 можно загрузить программы, написанные на языке БЕЙСИК.

ХОНИКС (машинные коды). Эта игра на логическое мышление и реакцию. Вам предстоит захватить как можно большую площадь игрового поля, избегая встреч с движущимися объектами.

СПИРОГРАФ (БЕЙСИК). Демонстрационная программа для цветного монитора.

КУРИЦА (ФОКАЛ). Игра вырабатывает быстроту реакции на случайные процессы.

БЛИНЫ (ФОКАЛ). Программа предлагает кулинарные хитрости выпечки блинов.

ТЕСТ 1, ТЕСТ 2 (ФОКАЛ). Психологический тест, позволяющий определить состояние человека и контролировать его изменения.

ПАКЕТ МЕДИЦИНСКИХ ПРОГРАММ (СПРАВКА, КОНТРЭКС 1А, КОНТРЭКС 2А, КОНТРЭКС 3, КУПЕР 1, КУПЕР 2, ТКП, РР1, РР2, РР4). Пакет демонстрирует возможность использования БК 0010 в медицинской диагностике (ФОКАЛ).

РАЛЛИ (машинные коды). Вы управляете рулем спортивного автомобиля. Попробуйте одолеть наибольшее количество препятствий при наивысшей скорости движения.

КОПИР (машинные коды). С помощью этой программы вы сможете быстро и качественно копировать ПС на БК-0010.

РЕД Т1 (машинные коды). Этот редактор текстов позволяет получать распечатки текстов на печатающее устройство УВВПЧ.

РУСЛАН (ФОКАЛ). Эта программа имитирует посадку самолета.

КРО (ФОКАЛ). С помощью этой программы вы сможете разместить буквы алфавита в определенной последовательности.

МАНЕВРЫ (ФОКАЛ). Вам предстоит управлять кораблем при проходе через минное поле.

ГРАФИКА (ФОКАЛ). Программа позволяет рисовать геометрические фигуры в графическом режиме с различной скоростью.

МАРАФОН (АССЕМБЛЕР). Вы примете участие в спасении людей во время военных действий.

САД (АССЕМБЛЕР). Вам предстоит собирать в саду фрукты, преодолевая различные препятствия.

ПИФ-ПАФ (АССЕМБЛЕР). Вы рыцарь, и вам предстоит справиться со сказочным чудовищем.

ЦЕНА РИСКА (АССЕМБЛЕР). С помощью этой программы вы сможете подняться по лестнице, перепрыгивая через препятствия.

СТРАНА (АССЕМБЛЕР). Эта программа предназначена для поиска кладов в лабиринте.

ШАХ (АССЕМБЛЕР). С помощью этой программы вы сможете сразиться в шахматы с компьютером.

МАРС (АССЕМБЛЕР). С помощью этой программы вам предстоит сразиться с воздушными завоевателями — «марсианами».

ЛАБИРИНТ (БЕЙСИК). Вам предстоит поиск сокровищ в лабиринтах пирамид.

АНТИКС (АССЕМБЛЕР). Это динамическая игра с летающими шарами.

КЛИНГ (БЕЙСИК). Вам предстоит нанести поражение вторгшимся в галактику звездным завоевателям — клингам.

Центры информатики ГКВТИ СССР

г. Москва,
Смоленский б-р, д. 4
тел. **246-19-23**

Ломоносовский пр., д. 34
тел. **143-85-22**

Фрунзенская наб., д. 50
тел. **242-88-65**

г. Иркутск,
ул. Лермонтова, д. 59
тел. **41-44-20**

г. Горький,
пер. Гоголя, д. 6а
тел. **34-19-08**

г. Казань,
пр. Х. Ямашева, д. 36
тел. **56-89-33**

г. Владивосток,
ул. Фокина, д. 18
тел. **6-54-87**

г. Куйбышев,
ул. Вольская, д. 13
тел. **51-19-41**

г. Саратов,
ул. Ленина, д. 161
тел. **64-55-40**

г. Баку,
ул. 28 апреля, д. 62
тел. **93-84-58**

г. Таллинн,
ул. Гоголя, д. 39
ул. Роона, д. 7А

г. Тбилиси,
пр. Дружбы, 3 кв., корп. 5
тел. **51-89-72**

г. Каунас,
ул. Траку, д. 4
тел. **20-77-07**

г. Бельцы,
ул. Стрыйская, д. 21
тел. **2-53-99**

г. Тирасполь,
пер. Красный, д. 2

г. Киев,
ул. Мельникова, д. 51
тел. **213-79-49**

ул. Строителей, д. 28/15
тел. **559-29-76**

г. Чернигов,
ул. Кирпоноса, д. 12
тел. **7-41-10**

г. Харьков,
пер. Нежинский, д. 9

г. Днепропетровск,
ул. Гоголя, д. 15

г. Ташкент,
ул. Чиланзар, кв. 1, д. 9

г. Воронеж,
пл. Детей, Дворец пионеров

г. Фрунзе,
ул. 50-летия Киргиз. ССР, д. 101
тел. **29-17-08**

г. Ереван,
сквер Абовяна, д. 1,
Дворец молодежи
тел. **53-01-83**

г. Омск,
ул. Гагарина, д. 24
тел. **33-04-08**

г. Георгиевск,
Ставропольский край,
ул. Октября, д. 140

г. Ленинград,
Петрозаводская сторона,
Большой пр-т, д. 18
тел. **235-06-18**

г. Минск,
ул. Славинского, д. 12
Дом технического творчества
тел. **64-60-17**

г. Рига,
ул. Плескалас, д. 1
тел. **26-07-00**

О. Я. БОКСЕР,
А. Г. ВАСИЛЬЧЕНКО,
А. А. ПРИЯТКИН

ГОДЕН, НЕ ГОДЕН (НУЖНОЕ ПОДЧЕРКНУТЬ)

В период всеобщей компьютеризации и перехода общества к информационной технологии становятся особенно актуальными задачи массовых измерений и исследования индивидуально-типологических характеристик человека. Это вызвано крайней важностью эффективного решения вопросов профотбора и профориентации, медицинской и социальной коррекции и т. д. [1]. Реакциометрия человека в данном случае выступает в качестве одного из фундаментальных направлений исследования человека. Ее результаты могут быть использованы для построения практических методик решения названных проблем.

В элементарном варианте (измерение так называемой простой реакции) реакциометрия человека сводится к следующему. Испытуемому дается инструкция: как только будет предъявлен какой-либо заранее обусловленный сигнал (например, световой или звуковой), необходимо как можно скорее воздействовать на датчик. При этом с высокой точностью измеряется время реакции.

На основе метода измерения простой реакции формируются методы изучения более сложных реакций различного уровня. При этом измеряется не только время реакции, но и ее своевременность (антиципирующая реакция на движущийся раздражитель, компенсаторное слежение и др.).

Современная реакциометрия позволяет тестировать с высокой достоверностью умственное и физическое утомление человека, степень воздействия на него стрессовых ситуаций, определять основные свойства его центральной нервной системы (типы высшей нервной деятельности). Особенно велико ее значение при изучении школьников в возрастном аспекте, спортсменов, летчиков, операторов атомных электростанций, диагностике психонервных заболеваний и др. [2].

В настоящее время советская и зарубежная промышленность производит специальные комплексы для измерения реакций, однако они выпускаются в крайне ограниченном количестве и высокой стоимости. Так, например, отечественный комплекс последнего поколения КПФО-02, созданный с участием одного из авторов данной статьи, выпускается в торговую сеть в количестве 30—40 штук в год и стоит 29 тыс. руб. Зарубежные комплексы, в частности австрийские, стоят в несколько раз дороже.

Широкое оснащение учебных заведений страны средствами электронной вычислительной техники как отдельными персональными ЭВМ (ПЭВМ), так и локальными сетями на их основе, позволяет решать вопросы массовой реакциометрии ресурсами учебных ПЭВМ, подобными комплектами типа КУВТ-86, «Агат», УКНЦ, «Корвет» и КУВТ — ЯМАХА. Этими ПЭВМ в настоящее время оснащено множество школ, ПТУ, вузов, и число их возрастает прогрессивно.

Распространение в учебных заведениях персональных ЭВМ, обладающих развитыми звуковыми, графическими, времяизмерительными и вычислительными ресурсами, возможностями передачи информации по локальной сети, а также сохранения ее на внешних запоминающих устройствах, открывает перспективу проведения массовых реакциометрических обследований на их основе.

Учебная ПЭВМ выступает здесь как универсальный инструмент реакциометрии человека. Ее использование в данном качестве существенно упрощает процесс тестирования с одновременным, практически неограниченным, возрастанием его функциональной эффективности.

Учебная ПЭВМ, применяемая в реакциометрических обследованиях, приводит к значительному сокращению материальных затрат в сравнении с традиционным использованием для этих целей специальных реакциометрических приборов, позволяет поднять практику профотбора на новый, качественно более высокий уровень. Здесь не требуются специально оборудованные лаборатории, высококвалифицированный персонал и дорогостоящие приборы. На учебные ПЭВМ массового применения возможно возложение функций хранения и корректировки физиологического паспорта человека.

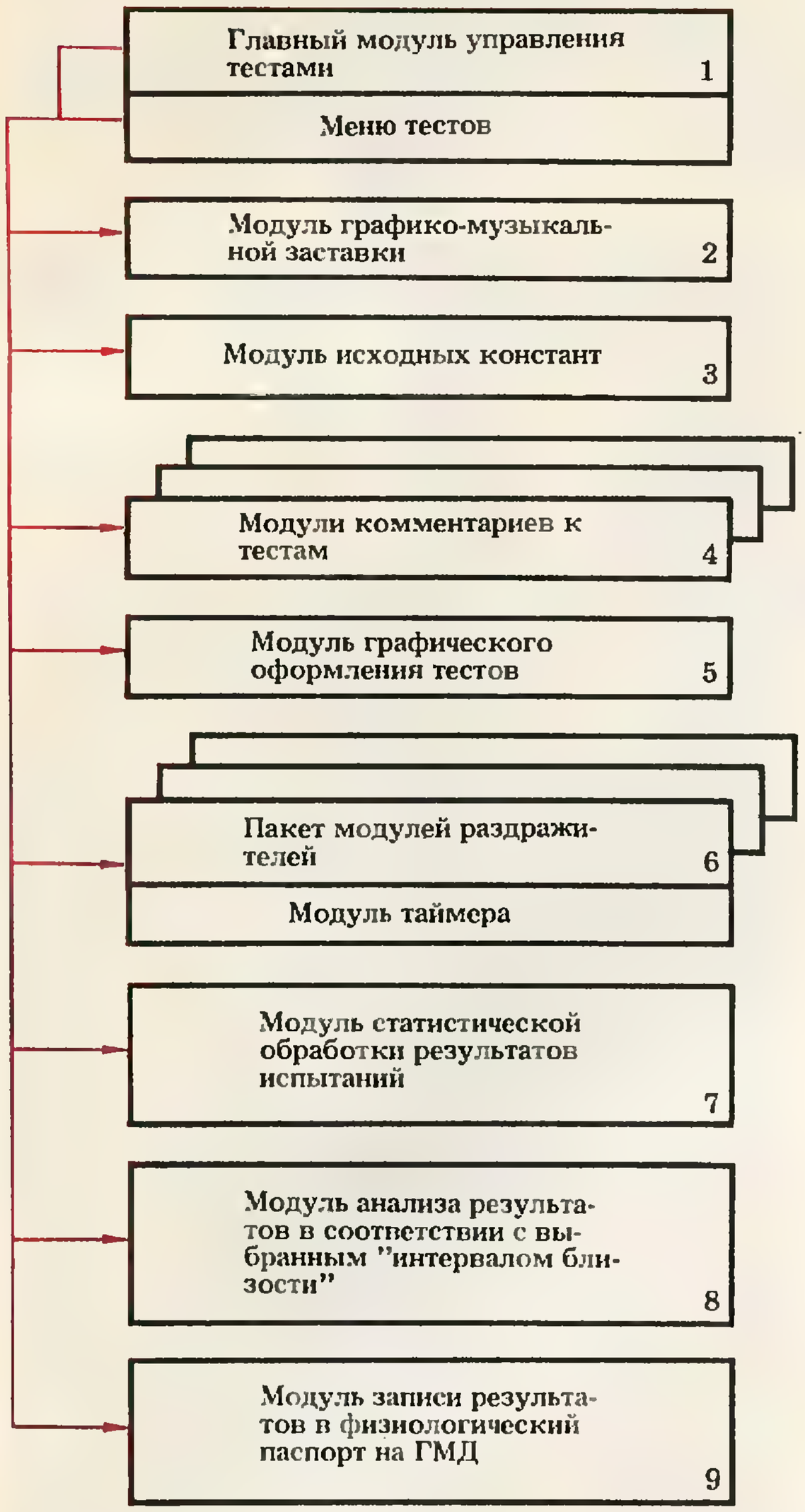
Таким образом, сравнительно простым путем, экономическое значение которого очевидно, полностью решается проблема внедрения реакциометрии в школу, институт, на производство, в спорт, учреждения здравоохранения и др.

В Шуйском государственном педагогическом институте на базе межкафедральной психофизиологической лаборатории ведутся работы по апробации учебных ПЭВМ в качестве реакциометрических приборов.

Разработанные авторами методики внедрены в ряде школ и спортивных секций города, в учебном процессе педагогического института. Ими легко овладевают не только студенты, но и школьники. За основу построения программного обеспечения взяты психофизиологические измерения, реализованные на лучших отечественных приборах «Символ-2МТ» и «Электроника НЦ-ТОНУС». При выборе учебных ПЭВМ ориентация производится на самые распространенные и дешевые ПЭВМ, такие, как «Электроника БК-0010Ш», «Ямаха-503» и др. [3]. В качестве базовых языков программирования используются наиболее популярные версии MSX-БЕЙСИК, TURBO-ПАСКАЛЬ, а также АССЕМБЛЕРы для реализации специальных функций приема и обработки информации. При разработке программного обеспечения используется структурный подход к архитектуре построения программ. Такой подход позволяет пользователям легко читать и модифицировать используемые программы, облегчает их верификацию [4].

Один из множества вариантов построения тест-программ реакциометрии на учебных ПЭВМ приведен на рис. 1. Тест-программа разбивается на ряд функционально законченных модулей, подчиняющихся иерархии управления, определенной главным модулем. В главный модуль, кроме меню тестов, вводятся также и различные сервисные возможности управления программой.

Каждый тест или программа в целом может на-



чинаться графико-музыкальной заставкой, отражающей характер теста, время и место его разработки. При необходимости на экран учебной ПЭВМ выводятся комментарии к тесту, определяющие порядок работы с ним. Для формирования исходных данных, вносимых в физиологический паспорт, используется модуль констант, в этом же модуле хранится информация о количестве испытаний в выборке, о законе распределения интервалов времени между испытаниями и т. п.

Модуль графического оформления теста выдает на экран ПЭВМ название теста, номер текущего испытания и другие характеристики, придавая выводимой информации наиболее приемлемый вид.

Описанные выше модули целесообразно разрабатывать на алгоритмических языках высокого уровня, что касается модулей раздражителей и связанного с ними таймера, то наиболее удачным способом следует признать их разработку на АССЕМБЛЕРах микропроцессора конкретной учебной ПЭВМ. Только таким способом возможно достижение необходимой контролируемой точности отсчета интервалов времени между моментом появления раздражителей на экране ПЭВМ (или в других его формах) и реакцией испытуемого на раздражитель [5].

Модуль статистической обработки результатов испытаний предназначен для приведения необходимого статистического анализа, результаты которого в последующем переписываются в физиологический паспорт испытуемого, хранимый на гибком магнитном диске.

Модуль анализа результатов испытаний в соответствии с выбранным интервалом близости предназначен для конечного анализа индивидуально-типологических характеристик с целью определения, например, профпригодности испытуемого.

Литература

1. Громов Г. Р. Гуманитарные основы информационной технологии. — М., 1988. — С. 36.

2. Боксер О. Я. Хронометраж физиологический // Большая медицинская энциклопедия. — М.: Наука, 1986. — 3-е. изд. — Т. 27. — С. 401—404.

3. Боксер О. Я. и др. Аппаратурно-программное обеспечение анализа психофизиологических функций рабочего в условиях производства. — ЦНТИ. — 1988.

4. Йордан Э. Структурное проектирование и конструирование программ. — М.: Мир, 1979.

5. Глушенко К. А. и др. Реализация функций измерения частоты в микропроцессорных системах // Микропроцессорные средства и системы. — 1988. — № 4. — С. 73—75.

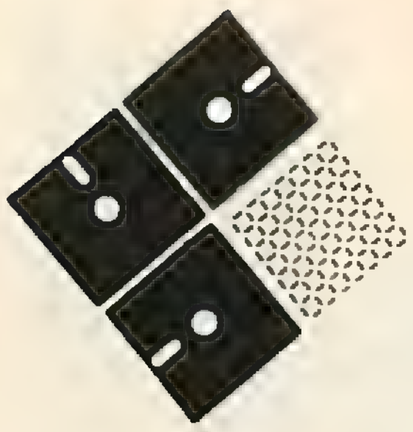
СОСТЯЗАНИЕ КОМПЬЮТЕРОВ

Состязание компьютеров состоялось в штате Иллинойс. В нем участвовало 100 ЭВМ, которым предстояло решить одинаковую задачу — систему линейных уравнений. По оценкам программистов, чтобы найти решение этих уравнений, компьютер должен был произвести около 700 тыс. арифметических операций.

В состязании 9 из 10 первых мест заняли компьютеры фирмы «Крэй», а лучший в Японии компьютер «НЕК—SX 2» оказался лишь на двенадцатом месте.

Победитель конкурса компьютер «YMR/832» фирмы «Крэй» нашел ответ за 0,003 с, тогда как компьютеры «ДЕК—ВАКС», «Сан» и «Аполло», которые широко применяются в США для инженерных расчетов, потратили на решение задачи от 5 до 10 с, а более дешевым компьютерам потребовалось от 1,5 до 3 мин.

Нью-Йорк (ТАСС),
25 января 1989 г.



Проблемы развития микроэлектроники входили составной частью в большинство национальных проектов, осуществлявшихся в Японии до настоящего времени, а некоторые проекты были целиком посвящены именно этой группе устройств информационной технологии и техники. Государство, располагающее передовой информационной технологией, владеет рынками сбыта, имеет авторитет, доходы и прекрасные перспективы для дальнейшего развития своего общества.

ЕСЛИ ПЛОХО СЕЙЧАС — НЕ ВСЕГДА ЖЕ ТАК БУДЕТ (ГОРАЦИЙ).

Японский стиль управления производством

После трех десятилетий феноменального роста Япония сегодня финансирует 10% мирового объема научных исследований и разработок и уступает только США по общему числу ученых - естествоиспытателей (чуть больше трети миллиона), по их числу на душу населения и по ежегодному числу оканчивающих высшие учебные заведения. Выпуск инженеров в Японии на 40% больше, чем в США, хотя население ее вдвое меньше...

С тех пор как превращение этой страны во вторую державу капиталистического мира стало свершившимся фактом, прошел уже не один год, а в западных исследованиях, посвященных этому явлению, все еще продолжают проскальзывать нотки удивления. Каким образом сумела она добиться таких результатов в столь короткие сроки? Ведь каких-то 15—20 лет назад никто всерьез не ожидал подобного рывка. Сегодня же можно нередко наблюдать обратную картину — США и страны Западной Европы пытаются перенять японский стиль управления производством.

Однако, по мере того как те же исследователи внимательно вглядываются в изменения, которые происходили в японской экономике за последние десятилетия, анализируют шаги, предпринимавшиеся правительством и ведущими промышленными фирмами по стимулированию отдельных направлений научно-технического прогресса, сравнивают их с действиями правительств и корпораций других государств, успехи Страны восходящего солнца быстро теряют ореол сенсации и представляются как вполне закономерный и заранее планировавшийся результат согласованного взаимодействия целого ряда факторов, как внутренних, так и внешних. Весомость каждого из них оценивается исследователями по-разному. Но в каком бы порядке различные авторы ни раскладывали набор причин, способствовавших достижениям японской экономики, все они на одно из первых мест ставят рациональность и эффективность мероприятий, входящих в понятие научно-технической политики, в частности так называемых национальных исследовательских программ, в рамках которых объединялись усилия государственного, частнопромышленного и академического секторов Японии.

Япония располагает минимумом собственных природных ресурсов и сильно зависит от других наций в обеспечении продуктами питания, энергией и сырьем. Поэтому одной из главных стратегических целей экономической политики Японии в послевоенный период является перестройка хозяйственной структуры страны в сторону наращивания удельного веса так называемых наукоемких отраслей и производств. Понимание этого привело к консенсусу между правительством, предпринимателями, финансовыми и академическими кругами по вопросу необходимости непрерывного наращивания национальной технологической базы.

На протяжении 60—70-х годов и в настоящее время большинство своих крупномасштабных программ разрабатывается на одном из самых приоритетных направлений научно-технического прогресса — в области информационной техники и технологии. Эта техника в работах японских авторов именуется «техникой трех К», т. е. компьютеры, коммуникация и контрольно-управляющая аппаратура, развивающиеся независимо друг от друга. Сегодня, главным образом благодаря быстрому прогрессу микроэлектроники, эти виды техники начали сливаться в единое целое и проникать во все области человеческой деятельности.

Япония не проводила и пока не проводит сколько-нибудь масштабных исследований военного характера, поскольку они официально запрещены конституцией страны. На оборону японцы тратили лишь порядка 0,5—0,7% общих расходов на науку. Промышленность в отношении исследований и разработок находится, так сказать, на самофинансировании; она потребляет даже несколько меньше, чем вносит, частично оплачивая исследования в университетах и бесприбыльных организациях.

Основная часть средств, выделяемых на науку государством, идет в университеты, и лишь небольшая часть используется для финансирования исследований и разработок в промышленности. При этом правительство никогда не распыляло свои дотации по всему фронту исследований, а всегда сосредоточивало их на основных, приоритетных направлениях, ориентируя соответствующим образом и частный сектор.

Электроника, а затем и весь комплекс информационной техники и технологии (ИТТ) на протяжении всего послевоенного периода относились к числу таких приоритетных направлений. Ему отводилась и отводится сегодня особая роль в структурной перестройке экономики в сторону наукоемкости. Объясняется это двумя причинами. Во-первых, все входящие в ИТТ отрасли сами по себе относятся к числу наукоемких, так что их интенсивное развитие вполне отвечает задачам упомянутой перестройки. Во-вторых, и это главное, информационная техника является своего рода преобразователем всех других отраслей хозяйства, как производственных, так и непроизводственных, основным средством их автоматизации, качественного изменения продукции и, как следствие, перевода частично или полностью в категорию наукоемких. Например, соединение традиционного станкостроения с информационной техникой превращают его в одну из наиболее сложных наукоем-

ких отраслей современной промышленности.

Основная функция японского правительства состоит в том, чтобы выбрать или организовать и проконтролировать выбор приоритетных технологий, уменьшить экономический риск, обычно ассоциируемый с разработкой новой техники, и помочь промышленным фирмам в обеспечении крупномасштабного производства. Прямая финансовая поддержка правительством исследований в приоритетных отраслях промышленности во многих случаях менее важна, чем сам факт причисления им (правительством) той или иной отрасли к числу приоритетных. Приоритетная отрасль приобретает престиж и общественное признание. Частные банки охотнее предоставляют ей долгосрочные кредиты, потребители и поставщики склонны оказывать ей предпочтение, чиновники в различных правительственных организациях более отзывчивы и внимательны к нуждам тех фирм, которые относятся к приоритетным отраслям.

Юридический статус одного из главных направлений в развитии экономики информационная техника и технология приобрели уже в 1957 г. Таким образом, постоянно существовала юридическая основа, обеспечивающая ИТТ благоприятные условия развития.

Еще один важный законодательный акт, на этот раз относящийся непосредственно к сфере исследований, был принят в 1962 г. Это так называемый закон о научно-технической кооперации в добывающей и обрабатывающей промышленности. Закон о кооперации открывал арсенал средств стимулирования промышленности и новые возможности. Он выводил объединения, создаваемые частными фирмами в целях проведения совместных исследовательских работ, из сферы действия антитрестовского законодательства. В других странах, особенно в США, такого рода объединения были практически противозаконны. С момента введения закона в действие кооперированные программы исследований становятся одной из основных форм развития научного потенциала страны, в том числе в области ИТТ.

Научно-технический потенциал Японии 60—70-х годов был ориентирован в первую очередь не на разработку оригинальных идей, а на изучение зарубежного опыта, отбор наиболее перспективных новинок, их освоение и усовершенствование, внедрение в производство и организацию масштабного выпуска новых видов продукции. Японцы выбрали для этого кратчайший и уже неоднократно опробованный в истории страны путь — заимствование и освоение новейших достижений путем закупки лицензий и ноу-хау, благо, западноевропейские и американские фирмы охотно продавали такие лицензии. При этом сравнительно низкая стоимость рабочей силы, а также до мелочей продуманная оптимальная технология обработки и сборки, тщательный контроль качества позволяли японским фирмам быстро выходить на рынок с конкурентоспособными ценами, зачастую стремительно вытесняя страны — прародительницы изобретений. За Японией прочно закрепилась репутация имитатора, пусть очень способного, но и только.

Все это, разумеется, правда. Но, как становится очевидным для многих исследователей при внимательном ретроспективном анализе, далеко не вся правда. Имитация имитации рознь. Можно слепо повторять образец, не внося в него никаких изменений и зачастую даже не до конца понимая, как он работает. Такие случаи в истории техники не редкость. Но можно имитировать и творчески, принимая образец лишь за основу, совершенствуя его в процессе воспроизводства, разрабатывая варианты и т. д. Как правило, японские специалисты шли вторым путем, так что из-под их рук выходили «копии», во многих отношениях превосходящие оригинал. При таком подходе накапливаются новыки и знания, научно-исследовательский потенциал, позволяющие в определенный момент естественно перейти от повторения к созданию качественно новых объектов.

Крупные открытия и изобретения в наши дни появляются вообще довольно редко, причем не столько как внезапное озарение гения-одиночки, сколько как результат длительной кропотливой работы целых коллективов, использующих очень сложную технику для проведения исследований. В «период имитации» такой потенциал у Японии накапливался. Да и без открытий не обходилось. Достаточно напомнить в этой связи, что волоконная оптика, положившая начало одному из важнейших направлений современной ИТТ, была изобретена в Японии.

Первый этап научно-технической политики завершается на рубеже 70—80-х годов. Япония

догнала передовые капиталистические страны по уровню экономического развития, а в ряде наукоемких областей обошла большинство из них. В полной мере это относится к ИТТ. Возможности имитационного пути уже в значительной степени исчерпаны. С одной стороны, японцам в ряде областей техники стало практически нечего покупать, а с другой — американцы и европейцы с каждым годом проявляют все меньше охоты продавать Японии информацию о своих достижениях и базовые патенты, защищающие новейшие разработки. Кроме того, японские фирмы, в ряде случаев пренебрегавшие фундаментальными исследованиями, имеют меньше новинок, способных заинтересовать западные компании, когда те запрашивают о возможностях перекрестного лицензирования.

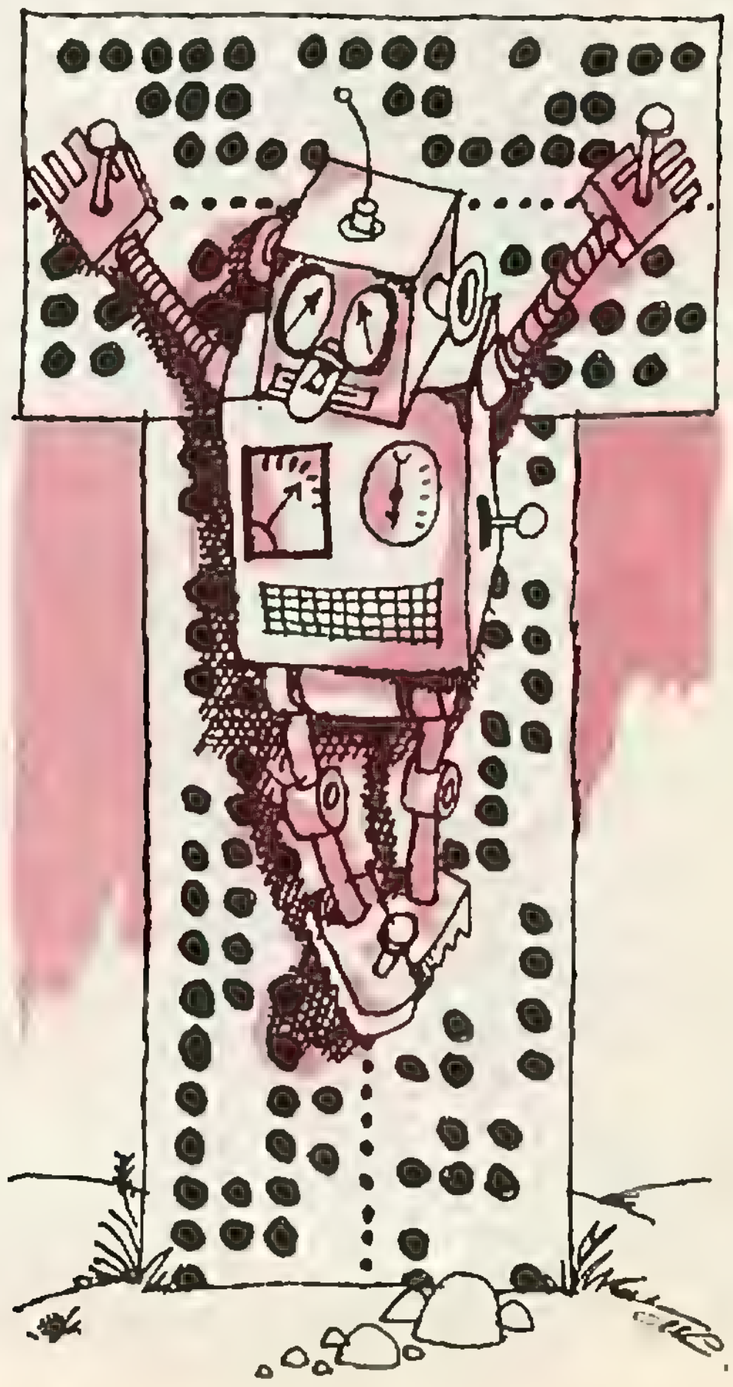
Итак, переориентация научно-технического потенциала на собственные оригинальные исследования и открытия — основная особенность нового, разворачивающегося в настоящее время этапа научно-технической политики Японии. Информационная техника продолжает оставаться в центре внимания. Информация лежит в основе современных тенденций развития, и Япония выбрала компьютеры и переработку информации в качестве главной национальной цели своего технического наступления.

Кооперация усилий государственного, частнопромышленного и академического секторов для проведения исследований, направленных на развитие информационной технологии, значительно расширяется. Из десяти проектов, находившихся на разных этапах реализации в 1986 г., шесть прямо относятся к ИТТ и охватывают широкий спектр проблем — от новейших методов изготовления интегральных схем до системы дистанционного зондирования поверхности Земли со спутников. Из 24 проектов (это, видимо, не полный перечень последних лет), опубликованных в литературе, приведены здесь только те, затраты на разработку которых оцениваются от 20 млн. иен и выше:

Название проекта	Сроки выполнения	Затраты млн. иен
Система обработки образной информации	1971—1980	22 000
Подводная добыча марганцевых конкреций	1981—1989	20 000
Вычислительная система для научных исследований	1981—1990	20 000
Передовая робототехника	1983—1990	20 000
Система обмена информацией между базами данных	1985—1991	20 000
Суперпрецизионная технология на основе использования энергетических пучков	1985—1993	30 000

Как видим, кроме одного, все перечисленные проекты относятся к информационной технике и технологии. Здесь нет мелкотемья, все проекты глобальны. Например, один из них — первый в перечне. В качестве главной технической цели проекта, как явствует из его названия, было выбрано создание современных систем распознавания образов, обработки и представления образной информации. Решение весьма ответственное, поскольку фактически речь шла об определении стратегической задачи важнейшей области техники на достаточно длительный период. Само содержание выбранной проблемы, по существу, охватывает весь спектр наиболее актуальных вопросов развития электронно-вычислительной техники 70-х годов — от разработки новой элементной базы до специализированных моделей ЭВМ различного назначения. Она же является одной из важнейших составных частей проблематики «искусственного интеллекта». Все это позволяет рассматривать обработку образной информации (проект PIPS) как ключевое звено развития электронно-вычислительной техники, способное в случае успеха потянуть за собою всю «компьютерную цепочку».

Отметим характерную для такого рода японских программ длительность подготовительного периода, занявшего примерно три года. И дело тут не только и даже не столько в сложности тематики или желании проработать ее как можно подробнее. Эту задачу удалось решить менее чем за год. Основное время ушло на неформальные консультации и обсуждения как внутри организаций, включившихся в подготовительный процесс, так и между ними, на достижение знаменитого японского консенсуса — и единства мнений всех участников по обсуждаемой проблеме, являющегося результатом приемлемых для них компромиссов. Иного пути, по сути дела, и не существует. Ведь административной власти и возможности решить вопрос приказом ни у одной из сторон практически нет. Японское правительство не управляет промышленными исследованиями и разработками путем финансирования их или в директивном порядке, скорее здесь существует взаимное партнерство между звеньями государствен-



ного аппарата и секторами промышленности, основанное на прагматических решениях, взаимном уважении, согласованной деятельности, направленной на достижение общих целей.

Характерным для большинства программ и проектов является преимущественное использование наиболее эффективного варианта кооперации — создание совместных коллективов исследователей, работающих «под одной крышей». Государство либо выделяет некоторой группе организаций (университетам, промышленным фирмам) определенные суммы денег, которые используются каждой из них для своих целей, либо выдает в той или иной форме заказы, направленные на достижение целей, отражающих интересы финансирующего государственного ведомства. Цель проекта у всех общая, для достижения ее все участники в той или иной степени объединяют свои усилия. Иначе, собственно, кооперация невозможна. Частные фирмы являются обязательными ее участниками, следовательно, объединяются конкуренты, которые за пределами совместного проекта жестко соперничают и на внутреннем рынке, и на мировом. Это предопределяет характер технических целей исследований — они всегда направлены не на создание конкретной законченной продукции, а на достижение так называемой доконкурентной стадии разработки. Тут могут быть новые физические эффекты и способы их использования, макеты, экспериментальные образцы и прототипы, в которых найдены и реализованы принципиальные технические решения, опробованы различные варианты и т. п.

Хотя внешне взаимоотношения между правительственным органом, участвующим в кооперации, и другими участниками выглядят обычно — правительство заключает с ними контракты на проведение определенных объемов исследований, по существу, ситуация существенно отличается от стандартной. Во-первых, контракты совершенно не означают, что министерство покупает или будет в последующем покупать результаты работы. Деньги, предоставляемые по контракту, — это форма субсидии, а не оплата услуг. Во-вторых, объем субсидии, как правило, не покрывает всей стоимости работ контрагента, значительную ее часть (иногда даже основную) он финансирует из собственного кармана. Японский исследователь Х. Ито называет получаемые фирмами от министерства субсидии «чисто символическими». В ряде случаев предусматривается даже частичный или полный возврат полученных фирмой-участником сумм, если она впоследствии сумеет выгодно использовать результаты совместной работы и извлечь из них прибыль. В то же время патенты, если таковые будут получены в ходе исследования, принадлежат государству. Участвующие в проекте фирмы получают лишь преимущественное, но не исключительное право приобретения такого патента. Таким образом, правительственное финансирование не является в данном случае главным мотивом, побуждающим промышленные корпорации к участию в разработках. Контракт служит лишь юридическим основанием, позволяющим министерству выполнять функции общего руководства проектом, придает ему (министерству) авторитет, необходимый для координации работ.

Однако сводить роль правительственного сектора лишь к общему руководству и координации (хотя и это весьма существенный вклад) было бы неверно. Кроме координации, он берет на себя две важные функции: определение целей и планирование работ и решение наиболее сложных фундаментальных проблем силами правительственных научных центров.

На основе тщательного всестороннего анализа состояния и тенденций развития японской и мировой науки, техники и экономики министерство внешней торговли и промышленности (MITI) составляет долгосрочные прогнозы, в соответствии с которыми формулируются национальные цели, а исходя из них — задачи конкретных национальных проектов. Прогнозы обычно охватывают десятилетний период и являются весьма авторитетными документами, поскольку оправдывались до настоящего времени с хорошей точностью. Что касается научного потенциала, которым располагает министерство, то ему подчинены в общей сложности 16 исследовательских институтов и организаций. При этом в одном из них — крупнейшем мировом центре электроники и вычислительной техники — электротехнической лаборатории (ETL) к началу 80-х годов работали около 700 человек, в том числе примерно 550 научных сотрудников. Годовой бюджет лаборатории равнялся 9000 млн. иен.

Трудно назвать область современной электроники, вычислительной техники, включая проблематику «искусственного интеллекта», оптоэлектроники или иных информационных технологий, которая осталась бы вне поля зрения исследователей ETL. Один пример — эта лаборатория первой в Японии занималась фундаментальным изучением транзисторов еще в то время, когда потенциальные технические возможности их использования только-только еще начинали осознаваться мировой наукой.

Два отмеченных выше момента — авторитет MITI в качестве «вперед смотрящего» японской промышленности и возможность прямого доступа к исследовательскому потенциалу ETL — являются если не решающими, то весьма важными факторами, побуждающими промышленные фирмы к участию в национальных проектах. А решающую роль играет объективная целесообразность кооперации, обусловленная сложностью стоящих перед современной информационной технологией проблем, требующих для своего решения очень больших капитальных вложений в исследования, сопровождающиеся немалым коммерческим риском. Академический сектор представлен не организациями, а лишь отдельными учеными, которых ETL привлекала к работе на различные сроки, выплачивая им соответствующие стипендии. Кроме того, в рамках проекта была организована так называемая «Программа исследователей-визитеров», предусматривавшая возможность приглашать зарубежных специалистов на конкретной основе для разработки отдельных конкретных тем. Всего по приглашениям в исследованиях приняли участие около 100 ученых-иностранцев.

И далее рассмотрение организационных вопросов исследований и разработок в Японии продолжим на примере системы образной информации (PIPS), наиболее крупной и завершенной на сегодняшний день.

Текущие задачи управления и координации решали главный администратор проекта, техниче-

ский менеджер и коллективный орган — собрание представителей участвующих в проекте организаций. В него входили 12 человек, по одному от каждой организации, председательствовал на собрании главный администратор. Такие собрания проходили три раза в год. Кроме того, при главном администраторе функционировал консультативный орган — комиссия крупномасштабного проекта.

Комиссия была наделена правом давать рекомендации главному администратору по вопросам окончательного выбора участников работы, распределении тематики, оценки результатов. Наконец, в управлении проектом участвовали два наблюдательных органа, являющихся подразделениями консультативного совета по промышленной технологии при министерстве.

Все участники исследований регулярно представляли отчеты о выполнении принятых на себя обязательств — полугодовые, годовые и итоговый отчет по теме. Эти отчеты являлись формой не только контроля, но и взаимной информации исследователей, так как материалы отчетов были доступны всем сотрудникам проекта. Второй этап работы над реализацией проекта PIPS начался весной 1977 г. и продолжался два года, до весны 1979 г. В нем участвовали уже только пять фирм — производителей электронных систем и ETL. На первом этапе в составе участников-исполнителей числились две исследовательские организации и десять промышленных фирм.

Дальнейшее «сплочение сил» характерно для следующего, третьего периода, занявшего 18 месяцев, с весны 1979 г. до конца лета 1980 г. Цель его — интегрировать разработанные подсистемы в единое целое, создать и отладить прототип новой ЭВМ. На это время была сформирована группа исследователей из 150 человек, которая работала в одном центре «Ikebukuro Sunshine building» в Токио. Правда, одновременно здесь трудилось обычно не более 80 человек, так как примерно половина членов группы командировалась для работы над проектом лишь в определенные дни недели. Заключительный, четвертый, этап проекта PIPS, продолжавшийся до марта 1981 г., был отведен для подведения итогов. Небольшая группа исследователей из 20 человек провела всесторонние испытания прототипа. Фирмы-участники кооперации и ETL составили подробные отчеты о всей проделанной работе. Результаты были рассмотрены и оценены как самой «Инженерной исследовательской ассоциацией PIPS», так и в консультативных и наблюдательных органах — комиссии крупномасштабного проекта и подкомитете оценки совета по промышленной технологии. Ассоциация и сам проект прекратили свое существование.

Организационные принципы, положенные в основу другой программы, начатой в 1981 г., — поисковые исследования в области новейших технологий (ERATO) — прямо противоположны традициям японской академической науки, да и не только академической. Ориентация на группу, подчинение интересов и действий личности интересам группы, которые ведут к подавлению инициативы, к стремлению «не слишком отличаться» от других, особое почтение к старшим по положению и по возрасту, вознаграждение не столько за творческий вклад, сколько за стаж, и тому подобные японские стереотипы — все это организаторами программы умышленно и решительно отвергается. По отношению к японским традициям программа ERATO является «ренегатом», поскольку в ней подчеркивается индивидуальность, а не группы, любознательность юности, а не мудрость и осторожность старости, интернационализм вместо чисто японского, замкнутого подхода. В работе над конкретным проектом, включенным в программу, должны участвовать примерно в равной пропорции исследователи не старше 35 лет из университетов, государственных научных учреждений и промышленных фирм. Приветствуется и поощряется привлечение ученых из зарубежных стран. Любой участник представляет не организацию, из которой он пришел, а только «самого себя и свои знания». С каждым заключается индивидуальный контракт, который подлежит ежегодному продлению, чтобы не было и намека на «пожизненный» найм. Директором проекта назначается известный ученый не старше 50 лет, и проект называют его именем. Для того чтобы коллектив был управляем, число его участников не должно превышать 30 человек. Длительность исследований ограничена сроком 5 лет, а объем затрат не должен превышать 2 млрд. иен, дабы избежать появления «бессмертных проектов», опасность которого в фундаментальной науке весьма реальна. В то же время для обеспечения «гибкости» перед проектом

предварительно никаких конкретных целей не ставится, они должны выкристаллизовываться по ходу работы.

В случае положительного решения проект утверждается, и вся дальнейшая организация работ предоставляется директору. По своему усмотрению (с учетом общих требований программы в отношении возраста и т. п.) он подбирает состав участников, разбивает общую тему на отдельные подтемы, составляет проект бюджета и согласовывает его с министерством финансов. В помощь себе он нанимает «административно-технический персонал проекта» — двух человек, одного технического специалиста и одного администратора.

Для изучения каждой подтемы директор формирует группу из нескольких (до пяти) человек и назначает руководителя группы. Группы работают раздельно, но раз в два-три месяца собираются для совместного обсуждения результатов. Руководители групп встречаются гораздо чаще. Ежегодно директор представляет в совет по исследованиям и разработкам промежуточные отчеты, а по окончании работ должен сдать подробный итоговый отчет о выполнении проекта.

Французский стиль управления производством

Советских производственников и ученых поражало, как рационально, продуманно и очень расчетливо французы подходят к проектам и организации работ, считая каждый франк. Основная проблема: кто и как будет финансировать работу, во что обойдется тот или иной проект. Технические трудности на втором плане. В противоположность у нас никогда не услышишь о финансовых трудностях, не узнаешь, сколько стоит тот или иной проект. Все вопросы об этом представители ведомств, Академии наук СССР просто игнорируют как «мелочь», предпочитая распространяться о технических трудностях и величинах решаемых задач.

Широко известны семейство французских ракет «Ариан», проект создания пилотируемого корабля многоразового использования «Гермес». Франция прилагает много усилий для развития национальной космической программы. Но посмотрите — реализацию проекта «Гермес» французы отдали в руки Европейского космического агентства. Тяжелую финансовую ношу выгоднее разложить на все страны «Общего рынка». А сотрудничество с СССР и США? Франция поддерживает добрые отношения и с той и с другой державой, но предпочтение отдает сотрудничеству с нами. Выгоднее. С США в основном оно идет на коммерческой основе, а с Советским Союзом преимущественно научное — без денежной компенсации.

Сейчас, когда уже близятся полеты «Гермеса», встал вопрос о создании Института космической медицины и физиологии. Организация работ в Институте в обязанности директора-организатора, который не хочет «изобретать велосипед». Если информация или разработка существуют и денежно доступны, мы ее купим и перепродадим. Если ее нет, закажем соответствующие исследования, конструкторские работы. Но не будем делать это сами. Зачем? Есть сеть специализированных государственных и коммерческих организаций, которые в состоянии уже сегодня выполнять наши заказы. Это быстрее и экономичнее.

Спонсорами института будут КНЕС — Национальный центр космических исследований, Центр медицинского здоровья и исследований, военно-воздушные силы.

В институте будут работать не более десяти человек. Директор, администратор, два секретаря, четыре специалиста для заключения договоров с промышленными фирмами, агентствами, научными организациями. Еще инженер и техник для конкретных оперативных работ.

В СССР в одном Институте медико-биологических проблем Минздрава СССР, специализирующемся в космической медицине, больше сотрудников, чем во всем Национальном центре космических исследований Франции. А ведь КНЕС занимается формированием программ, фундаментальными научными исследованиями. И всего на начало 1988 г. здесь были заняты 2227 человек. У нас, кажется, больше в одном только Институте космических исследований Академии наук СССР.

Во Франции передачей космических достижений народному хозяйству занимаются акционерные коммерческие фирмы. Одна из них — «Навспейс» с начальным капиталом 5 млн. франков (~500 тыс. рублей) имеет десять сотрудников.

Чтобы конкурировать с западными фирмами, и нам придется считать каждый рубль, придется решать задачи «не числом, а уменьем». Ни для кого не секрет, что работа с искусственно раздутыми штатами сопровождается организационным беспорядком (а может быть, и хаосом) и большими экономическими потерями. И все же, если плохо сейчас, не всегда же так будет.

По материалам научно-аналитического обзора (ИНИОН) и газеты «Известия» (28 декабря 1988 г.)

Что взамен структуры фон Неймана? — М.: Знание, Ч80 1989. — 48 с. — (Новое в жизни, науке, технике. Сер. «Вычислительная техника и ее применение», № 5).

ISBN 5-07-000483-2

20 к.

Разработка вычислительной техники новых поколений и соответственно разработка новых архитектурных решений ЭВМ представляют собой одну из актуальных и неотложных задач науки и производства. В сборнике рассказывается об основных проблемах создания и разработки современных ЭВМ.

Брошюра рассчитана на широкий круг читателей.

2404000000 **ББК 32.97**

ТЕМА *СЛЕДУЮЩЕГО* **НОМЕРА:**

РАДИО-ЭЛЕКТРОНИКА И СВЯЗЬ

Новые виды электросвязи (Электронная почта)

***ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА* И ЕЁ ПРИМЕНЕНИЕ**

Сюита для персонального компьютера с оркестром

МАТЕМАТИКА КИБЕРНЕТИКА

Многокритериальное проектирование машин

Научно-популярное издание

ЧТО ВЗАМЕН СТРУКТУРЫ фон НЕЙМАНА?

Гл. отраслевой редактор *Л. А. Ерлыкин*
Зам. гл. отраслевого редактора *Г. Г. Карвовский*
Редактор *Б. М. Васильев*
Мл. редактор *Н. А. Васильева*
Художник *В. Н. Конюхов*
Худож. редактор *М. А. Гусева*
Техн. редактор *А. М. Красавина*
Корректор *Л. С. Соколова*

ИБ № 10132

Сдано в набор 24.01.89. Подписано к печати 27.03.89. Т-00920. Формат бумаги 70×100$^{1}/_{16}$. Бумага офсетная № 2. Гарнитура журнально-рубленая. Печать офсетная. Усл. печ. л. 3,90. Усл. кр.-отт. 8,45. Уч.-изд. л. 4,98. Тираж 68 162 экз. Заказ 2978. Цена 20 коп. Издательство «Знание». 101835, ГСП, Москва, Центр, проезд Серова, д. 4. Индекс заказа 894705.
Ордена Трудового Красного Знамени Калининский полиграфический комбинат Государственного комитета СССР по делам издательств, полиграфии и книжной торговли. 170024, г. Калинин, пр. Ленина, 5.

Цена 20 коп.

Индекс 70195

Адрес подписчика:

Лен. 27-43

Издательство «Знание»

Подписная научно-популярная серия

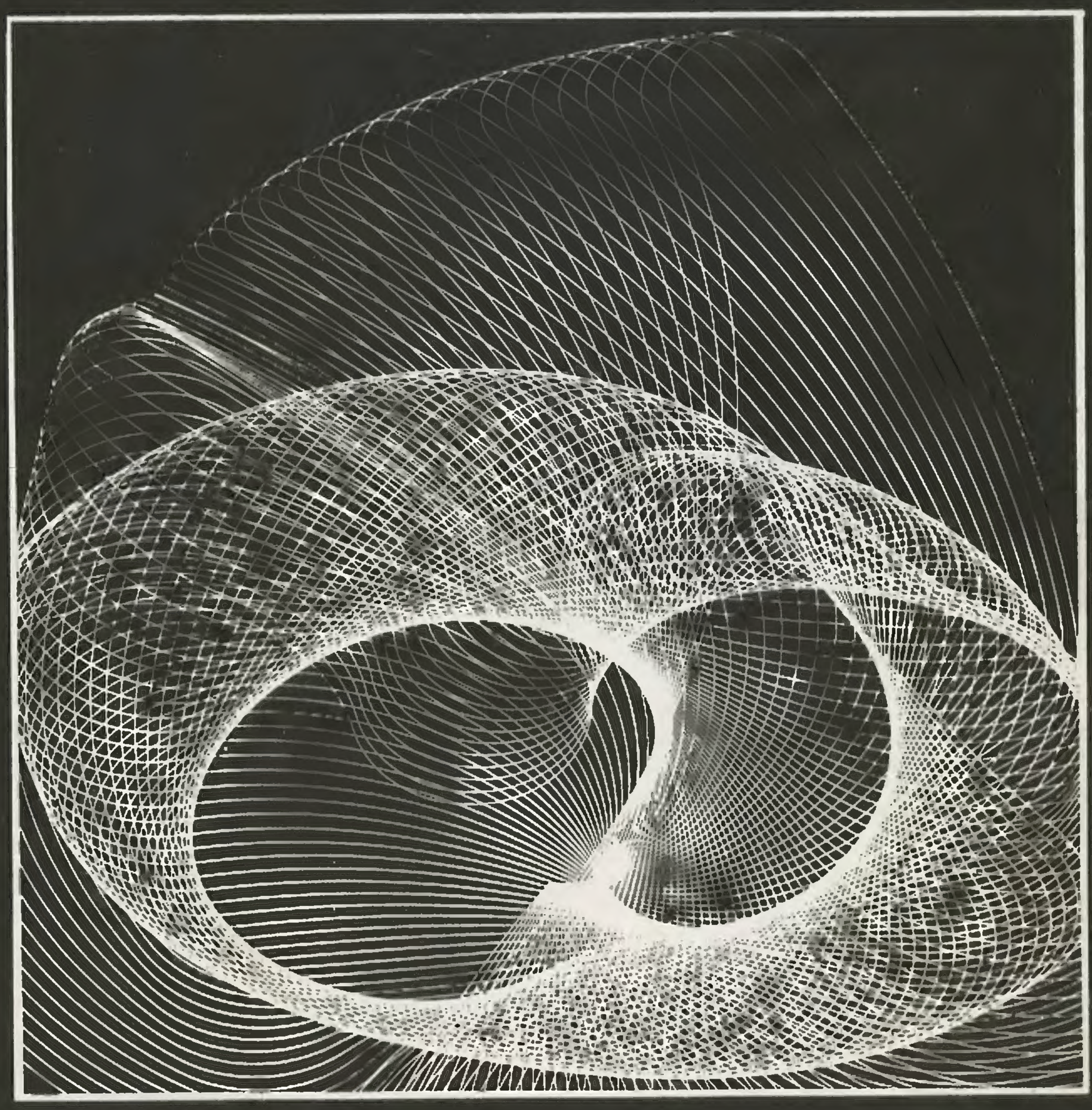
ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Компьютер — "Он не может быть добрым, дружелюбным, изобретательным, иметь чувство юмора, отличать истинное от ложного, делать ошибки, влюбляться и становиться предметом любви, получать наслаждение от клубники со сливками, правильно пользоваться словами и размышлять о своей судьбе, совершать нечто действительно новое и быть столь же разнообразным, как и человек".

А. Тьюринг

"Вычислительная машина и ценна лишь настолько, насколько ценен использующий ее человек".

Н. Винер



Наш адрес:
СССР,
Москва,
Центр,
проезд
Серова, 4